

#3

Docket No. 1075.1137/JDH

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of: )  
 )  
Masayoshi HASHIMA, et al. )  
 ) Group Art Unit: Unassigned  
Serial No.: New )  
 ) Examiner: Unassigned  
Filed: January 3, 2001 )  
 )  
For: SUPPORT SYSTEM, AND COMPUTER-)   
READABLE RECORDING MEDIUM )  
IN WHICH SUPPORT PROGRAM )  
IS RECORDED )

JCS15 U.S. PRO  
09/754301  
01/05/01

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN  
APPLICATION IN ACCORDANCE  
WITH THE REQUIREMENTS OF 37 C.F.R. §1.55**

*Honorable Commissioner of  
Patents and Trademarks  
Washington, D.C. 20231*

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-034642  
Filed: February 14, 2000.

It is respectfully requested that the applicants be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,  
STAAS & HALSEY LLP

Date: January 3, 2001

By:

James D. Halsey, Jr.  
Registration No. 22,729

700 Eleventh Street, N.W., Suite 500  
Washington, D.C. 20001  
(202) 434-1500

日 本 国 特 許 庁  
PATENT OFFICE  
JAPANESE GOVERNMENT

1c915 U.S. PTO  
09/754301  
01/05/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日  
Date of Application:

2000年 2月14日

出 願 番 号  
Application Number:

特願2000-034642

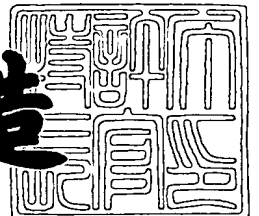
出 願 人  
Applicant(s):

富士通株式会社

2000年 9月29日

特許庁長官  
Commissioner,  
Patent Office

及 川 耕 造



出証番号 出証特2000-3080248

【書類名】 特許願

【整理番号】 9951877

【提出日】 平成12年 2月14日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/50

【発明の名称】 支援システムおよび組込み支援プログラムを記録したコンピュータ読み取り可能な記録媒体

【請求項の数】 18

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 橋間 正芳

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 千田 陽介

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 佐藤 裕一

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100092978

【弁理士】

【氏名又は名称】 真田 有

【電話番号】 0422-21-4222

【手数料の表示】

【予納台帳番号】 007696

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9704824

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 支援システムおよび組込み支援プログラムを記録したコンピュータ読み取り可能な記録媒体

【特許請求の範囲】

【請求項 1】 アクチュエータおよびセンサを含む複数の部品からなる機構を 3 次元的に設計する機構設計部と、

該機構を 3 次元機構モデルとして内部に構築され該機構の動作をシミュレートする 3 次元機構モデルシミュレーション部と、

該機構に組み込まれ該機構の動作を制御するための制御プログラムを組込みソフトウェアとして開発する組込みソフトウェア開発部と、

該機構設計部による設計データを、該 3 次元機構モデルに反映させるべく該機構設計部から該 3 次元機構モデルシミュレーション部へ入力する第 1 インタフェース部と、

該 3 次元機構モデルシミュレーション部の動作と該組込みソフトウェア開発部の動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該組込みソフトウェア開発部との間でデータ送受を行なう第 2 インタフェース部とをそなえて構成されていることを特徴とする、支援システム。

【請求項 2】 該第 1 インタフェース部が、該 3 次元機構モデルシミュレーション部によるシミュレーション結果を、該機構の設計に反映させるべく該 3 次元機構モデルシミュレーション部から該機構設計部へ入力することを特徴とする、請求項 1 記載の支援システム。

【請求項 3】 該組込みソフトウェア開発部が、該組込みソフトウェアの詳細設計を行なうべく該組込みソフトウェアの仕様を記述する状態遷移図・表を作成・編集する状態遷移図・表作成部を含んで構成され、

該第 2 インタフェース部が、該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該状態遷移図・表作成部との間でデータ送受を行なうことを特徴とする、請求項 1 または請求項 2 に記載の支援システム。

【請求項 4】 該状態遷移図・表作成部が、複数のタスクを並行して実行す

るマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、

該第 2 インタフェース部が、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させることを特徴とする、請求項 3 記載の支援システム。

【請求項 5】 該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させることを特徴とする、請求項 4 記載の支援システム。

【請求項 6】 該組込みソフトウェア開発部が、開発中の該組込みソフトウェアを組み込まれたマイクロコンピュータチップを含んで構成され、

該第 2 インタフェース部が、該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該マイクロコンピュータチップとの間でデータ送受を行なうことを特徴とする、請求項 1～請求項 5 のいずれか 1 項に記載の支援システム。

【請求項 7】 該マイクロコンピュータチップが、複数のタスクを並行して実行するマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、

該第 2 インタフェース部が、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させることを特徴とする、請求項 6 記載の支援システム。

【請求項 8】 該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させることを特徴とする、請求項 7 記載の支援システム。

【請求項 9】 該第 2 インタフェース部が、

該組込みソフトウェア開発部から該 3 次元機構モデルシミュレーション部へ、  
該 3 次元機構モデルにおける該アクチュエータに対するアクチュエータ指令信号  
を受け渡すとともに、

該 3 次元機構モデルシミュレーション部から該組込みソフトウェア開発部へ、  
該アクチュエータ指令信号に応じたシミュレーション結果として得られる、該 3  
次元機構モデルにおける該センサからのセンサ信号を受け渡すことを特徴とする  
、請求項 1 ～請求項 8 のいずれか 1 項に記載の支援システム。

【請求項 1 0】 該アクチュエータに対する該アクチュエータ指令信号と該  
3 次元機構モデルシミュレーション部からの該センサ信号との時間変化をリアル  
タイムで解析して表示しうる解析部をそなえて構成されていることを特徴とする  
、請求項 9 記載の支援システム。

【請求項 1 1】 アクチュエータおよびセンサを含む複数の部品からなる機  
構を制御すべくその機構に制御プログラムとして組み込まれる組込みソフトウェ  
アの開発を支援する機能をコンピュータにより実現するための、支援プログラム  
を記録したコンピュータ読み取り可能な記録媒体であって、

該支援プログラムとして、

該機構を 3 次元的に設計する機構設計部として、該コンピュータを機能させる  
機構設計プログラムと、

該機構を 3 次元機構モデルとして内部に構築され該機構の動作をシミュレート  
する 3 次元機構モデルシミュレーション部として、該コンピュータを機能させる  
3 次元機構モデルシミュレーションプログラムと、

該組込みソフトウェアを開発する組込みソフトウェア開発部として、該コンピ  
ュータを機能させる組込みソフトウェア開発プログラムと、

該機構設計部による設計データを、該 3 次元機構モデルに反映させるべく該機  
構設計部から該 3 次元機構モデルシミュレーション部へ入力する第 1 インタフェ  
ース部として、該コンピュータを機能させる第 1 インタフェースプログラムと、

該 3 次元機構モデルシミュレーション部の動作と該組込みソフトウェア開発部  
の動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該組込み  
ソフトウェア開発部との間でデータ送受を行なう第 2 インタフェース部として、

該コンピュータを機能させる第2インタフェースプログラムとが記録されていることを特徴とする、支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項12】 該第1インタフェースプログラムが、該3次元機構モデルシミュレーション部によるシミュレーション結果を、該機構の設計に反映させるべく該3次元機構モデルシミュレーション部から該機構設計部へ入力させるように構成されていることを特徴とする、請求項1記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項13】 該組込みソフトウェア開発プログラムが、該組込みソフトウェアの詳細設計を行なうべく該組込みソフトウェアの仕様を記述する状態遷移図・表を作成・編集する状態遷移図・表作成部として、該コンピュータを機能させる状態遷移図・表作成プログラムを含むものであり、

該第2インタフェースプログラムが、該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作との同期をとりながら該3次元機構モデルシミュレーション部と該状態遷移図・表作成部との間でデータ送受を行なわせるように構成されていることを特徴とする、請求項11または請求項12に記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項14】 該状態遷移図・表作成プログラムが、複数のタスクを並行して実行するマルチタスクを採用し、該3次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行させるように構成され、

該第2インタフェースプログラムが、該同期タスクを用いて該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させることを特徴とする、請求項13記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項15】 該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させることを特徴とする、請求項14記載の支援プログラムを記録したコンピュータ読み取り可能な記録



媒体。

【請求項 1 6】 該第 2 インタフェースプログラムが、該 3 次元機構モデルシミュレーション部の動作と、開発中の該組込みソフトウェアを組み込まれたマイクロコンピュータチップの動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該マイクロコンピュータチップとの間でデータ送受を行なわせるように構成されていることを特徴とする、請求項 1 1 ～請求項 1 5 のいずれか 1 項に記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項 1 7】 該マイクロコンピュータチップが、複数のタスクを並行して実行するマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、

該第 2 インタフェースプログラムが、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させることを特徴とする、請求項 1 6 記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項 1 8】 該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させることを特徴とする、請求項 1 7 記載の支援プログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、アクチュエータやセンサを有し 3 次元的な動作を行なう機構の設計・開発にかかる技術であって、特に、機構を制御すべくその機構に組み込まれる制御プログラム（組込みソフトウェア）の開発を支援するための、支援システムおよび支援プログラムを記録したコンピュータ読み取り可能な記録媒体に関する。

【0 0 0 2】

## 【従来の技術】

一般に、アクチュエータ（モータ）やセンサを有し 3 次元的な動作を行なう機構（メカ）を設計する際には、その機構の構想を練った後、詳細設計、出図、部品手配を行なってから、部品の組立を行なって実機を試作し、実機の動作等についての評価を行なう。そして、評価の結果に応じて設計変更を行なってから設計変更後の実機を試作し、再び評価を行なうという処理を繰り返し、評価の結果が良好であれば、設計を完了する。なお、ここでいう機構としては、例えば CD チェンジャ、MD チェンジャ、プリンタ、マニピュレータ等が挙げられる。

## 【0003】

また、一般に、上述のごとく設計された機構を動作させるべく、その機構を制御するための制御プログラムを開発し、その制御プログラムを、制御対象の機構内に組み込まれるマイクロコンピュータ（以下、マイコンという場合がある）で実行させるようにしている。このマイコンが実行する上記制御プログラムのことを、以下、組込みソフトウェアという場合がある。

## 【0004】

ところで、上述のような制御プログラム（組込みソフトウェア）を開発する際には、従来、制御すべき機構の試作品（実機）が完成している必要がある。即ち、試作が完了して初めてメカを具体的に動かすことができ、それを使って組込みソフトウェアの開発を開始することができるわけである。

この組込みソフトウェアの開発は、試作品の完成後、その試作品を実際に動作させながら、以下の手順で行なわれる。つまり、まず、組込みソフトウェアの概略設計を行ない、その概略設計に基づいて詳細設計を行なってから、詳細設計の結果をコーディングして組込みソフトウェアを作成し、その組込みソフトウェアのデバッグを行なう。

## 【0005】

## 【発明が解決しようとする課題】

しかしながら、上述のように、組込みソフトウェア（制御プログラム）の開発を行なう際には制御対象の機構の試作品（実機）が必要であるため、試作品が完成するまでは組込みソフトウェアの開発を開始することができず、その開発に時

間やコストがかかり非効率的であるという課題があった。

【 0 0 0 6 】

また、試作品が完成して組込みソフトウェアの開発を開始したとしても、機構の試作段階では評価結果に応じて試作品の設計変更を行なうことは度々あり、その都度、組込みソフトウェアの開発を中断して、設計変更後の試作品の完成を待ってから、設計変更後の試作品に応じた仕様変更や再コーディングを行なわなければならない、その開発にさらなる時間やコストがかかり極めて非効率的であるという課題もあった。

【 0 0 0 7 】

このように、従来、機構の設計（実機の作成）とその機構のための組込みソフトウェアの開発とのコンカレント性は全くなく、同時に実行することができなかったもので、組込みソフトウェアの開発は極めて非効率的に行なわれている。このため、機構の設計と組込みソフトウェアの開発とをコンカレントに（並行して）遂行できるようにすることが望まれている。

【 0 0 0 8 】

本発明は、このような課題に鑑み創案されたもので、機構の設計とその機構を制御するための組込みソフトウェアの開発とをコンカレントに遂行できるようにして、その組込みソフトウェアの開発の効率化をはかり、開発期間の短縮，工数の削減を実現した、支援システムおよび支援システムを記録したコンピュータ読み取り可能な記録媒体を提供することを目的とする。

【 0 0 0 9 】

【課題を解決するための手段】

上記目的を達成するために、本発明の支援システム（請求項 1）は、アクチュエータおよびセンサを含む複数の部品からなる機構を 3 次元的に設計する機構設計部と、該機構を 3 次元機構モデルとして内部に構築され該機構の動作をシミュレートする 3 次元機構モデルシミュレーション部と、該機構に組み込まれ該機構の動作を制御するための制御プログラムを組込みソフトウェアとして開発する組込みソフトウェア開発部と、該機構設計部による設計データを該 3 次元機構モデルに反映させるべく該機構設計部から該 3 次元機構モデルシミュレーション部へ

入力する第1インタフェース部と、該3次元機構モデルシミュレーション部の動作と該組込みソフトウェア開発部の動作との同期をとりながら該3次元機構モデルシミュレーション部と該組込みソフトウェア開発部との間でデータ送受を行なう第2インタフェース部とをそなえて構成されていることを特徴としている。

【0010】

このとき、該第1インタフェース部が、該3次元機構モデルシミュレーション部によるシミュレーション結果を、該機構の設計に反映させるべく該3次元機構モデルシミュレーション部から該機構設計部へ入力してもよい（請求項2）。

また、該組込みソフトウェア開発部が、該組込みソフトウェアの詳細設計を行なうべく該組込みソフトウェアの仕様を記述する状態遷移図・表を作成・編集する状態遷移図・表作成部を含み、該第2インタフェース部が、該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作との同期をとりながら該3次元機構モデルシミュレーション部と該状態遷移図・表作成部との間でデータ送受を行なうように構成してもよい（請求項3）。

【0011】

この場合、該状態遷移図・表作成部が、複数のタスクを並行して実行するマルチタスクを採用し、該3次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、該第2インタフェース部が、該同期タスクを用いて該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させてもよい（請求項4）。そして、該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該3次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させるように構成してもよい（請求項5）。

【0012】

さらに、該組込みソフトウェア開発部が、開発中の該組込みソフトウェアを組み込まれたマイクロコンピュータチップを含み、該第2インタフェース部が、該3次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作との同期をとりながら該3次元機構モデルシミュレーション部と該マイクロ

コンピュータチップとの間でデータ送受を行なうように構成してもよい（請求項 6）。

【0013】

この場合、該マイクロコンピュータチップが、複数のタスクを並行して実行するマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、該第 2 インタフェース部が、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させてもよい（請求項 7）。そして、該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させるように構成してもよい（請求項 8）。

【0014】

なお、該第 2 インタフェース部が、該組込みソフトウェア開発部から該 3 次元機構モデルシミュレーション部へ、該 3 次元機構モデルにおける該アクチュエータに対するアクチュエータ指令信号を受け渡すとともに、該 3 次元機構モデルシミュレーション部から該組込みソフトウェア開発部へ、該アクチュエータ指令信号に応じたシミュレーション結果として得られる、該 3 次元機構モデルにおける該センサからのセンサ信号を受け渡すように構成してもよい（請求項 9）。

【0015】

また、該アクチュエータに対する該アクチュエータ指令信号と該 3 次元機構モデルシミュレーション部からの該センサ信号との時間変化をリアルタイムで解析して表示しうる解析部をそなえてもよい（請求項 10）。

一方、本発明の記録媒体（請求項 11）は、アクチュエータおよびセンサを含む複数の部品からなる機構を制御すべくその機構に制御プログラムとして組み込まれる組込みソフトウェアの開発を支援する機能をコンピュータにより実現するための、支援プログラムを記録したコンピュータ読み取り可能なものであって、該支援プログラムとして、①該機構を 3 次元的に設計する機構設計部として、該コンピュータを機能させる機構設計プログラムと、②該機構を 3 次元機構モデル

として内部に構築され該機構の動作をシミュレートする 3 次元機構モデルシミュレーション部として、該コンピュータを機能させる 3 次元機構モデルシミュレーションプログラムと、③該組込みソフトウェアを開発する組込みソフトウェア開発部として、該コンピュータを機能させる組込みソフトウェア開発プログラムと、④該機構設計部による設計データを、該 3 次元機構モデルに反映させるべく該機構設計部から該 3 次元機構モデルシミュレーション部へ入力する第 1 インタフェース部として、該コンピュータを機能させる第 1 インタフェースプログラムと、⑤該 3 次元機構モデルシミュレーション部の動作と該組込みソフトウェア開発部の動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該組込みソフトウェア開発部との間でデータ送受を行なう第 2 インタフェース部として、該コンピュータを機能させる第 2 インタフェースプログラムとを記録したことを特徴としている。

## 【 0 0 1 6 】

このとき、該第 1 インタフェースプログラムが、該 3 次元機構モデルシミュレーション部によるシミュレーション結果を、該機構の設計に反映させるべく該 3 次元機構モデルシミュレーション部から該機構設計部へ入力させるように構成されてもよい（請求項 1 2）。

また、該組込みソフトウェア開発プログラムが、該組込みソフトウェアの詳細設計を行なうべく該組込みソフトウェアの仕様を記述する状態遷移図・表を作成・編集する状態遷移図・表作成部として、該コンピュータを機能させる状態遷移図・表作成プログラムを含むものであり、該第 2 インタフェースプログラムが、該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該状態遷移図・表作成部との間でデータ送受を行なわせるように構成されてもよい（請求項 1 3）。

## 【 0 0 1 7 】

この場合、該状態遷移図・表作成プログラムが、複数のタスクを並行して実行するマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを

該複数のタスクとは別に実行させるように構成され、該第 2 インタフェースプログラムが、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させてもよい（請求項 1 4）。そして、該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該状態遷移図・表作成部の動作とを同期させてもよい（請求項 1 5）。

【0 0 1 8】

さらに、該第 2 インタフェースプログラムが、該 3 次元機構モデルシミュレーション部の動作と、開発中の該組込みソフトウェアを組み込まれたマイクロコンピュータチップの動作との同期をとりながら該 3 次元機構モデルシミュレーション部と該マイクロコンピュータチップとの間でデータ送受を行なわせるように構成されてもよい（請求項 1 6）。

【0 0 1 9】

この場合、該マイクロコンピュータチップが、複数のタスクを並行して実行するマルチタスクを採用し、該 3 次元機構モデルシミュレーション部のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを該複数のタスクとは別に実行し、該第 2 インタフェースプログラムが、該同期タスクを用いて該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させてもよい（請求項 1 7）。そして、該同期タスクの優先度を最も高く設定し、該同期タスクにより該複数のタスクの開始・停止を制御して、該 3 次元機構モデルシミュレーション部の動作と該マイクロコンピュータチップの動作とを同期させてもよい（請求項 1 8）。

【0 0 2 0】

【発明の実施の形態】

以下、図面を参照して本発明の実施の形態を説明する。

図 3 は、本発明の一実施形態としての支援システムを構築しうるコンピュータシステムを示す外観図である。

この図 3 に示すように、コンピュータシステム 1 0 0 は、図 4 を参照しながら後述するごとく CPU 2 2 1、RAM 2 2 2、ハードディスク 2 1 1 等を内蔵し

た本体部 1 0 1 と、この本体部 1 0 1 からの指示により表示画面 1 0 2 a で画面表示を行なう C R T ディスプレイ 1 0 2 と、このコンピュータシステム 1 0 0 にユーザの指示や文字情報を入力するためのキーボード 1 0 3 と、表示画面 1 0 2 a 上の任意の位置を指定することによりその位置に表示されているアイコン等に応じた指示を入力するマウス 1 0 4 とをそなえて構成されている。

【 0 0 2 1 】

本体部 1 0 1 は、さらに、外観上、フロッピーディスク 2 1 2 (図 4 参照) や C D - R O M 2 1 0 がそれぞれ取り出し自在に装填されるフロッピーディスク装填口 1 0 1 a および C D - R O M 装填口 1 0 1 b を有しており、その内部には、装填されたフロッピーディスク 2 1 2 や C D R O M 2 1 0 をドライブする、フロッピーディスクドライバ 2 2 4 や C D R O M ドライバ 2 2 5 (図 4 参照) を有している。

【 0 0 2 2 】

ここでは、C D - R O M (コンピュータ読み取り可能な記録媒体) 2 1 0 に、本発明の一実施形態としての支援プログラム 3 0 0 (図 2 参照) が記録されている。

この C D - R O M 2 1 0 が C D - R O M 装填口 1 0 1 b から本体部 1 0 1 内に装填され、C D - R O M ドライバ 2 2 5 により、その C D - R O M 2 1 0 に記録された支援プログラム 3 0 0 がこのコンピュータシステム 1 0 0 のハードディスク 2 1 1 内にインストールされる。

【 0 0 2 3 】

また、コンピュータシステム 1 0 0 のハードディスク 2 1 1 には、図 1 を参照しながら後述するメカ設計部 (移送設計・3 次元 C A D システム) 2 0 により設計されて定義された、設計対象機構についての設計データが格納され、その設計データに基づき、コンピュータシステム 1 0 0 (後述する 3 次元リアルタイムシミュレーション装置 3 0) において、その機構についての 3 次元機構モデルが構築されるようになっている。

【 0 0 2 4 】

さらに、コンピュータシステム 1 0 0 のハードディスク 2 1 1 には、図 1 を参



照しながら後述する組込みソフトウェア開発部 4 0 により開発される制御プログラム（組込みソフトウェア）が格納される。ハードディスク 2 1 1 には、開発を完了した組込みソフトウェアが格納されるほか、開発中の組込みソフトウェアも格納・保存される。

#### 【 0 0 2 5 】

図 4 は、本発明の一実施形態としての支援システムを構築しうるコンピュータシステムのハードウェア構成を示すブロック図であり、この図 4 に示すように、本実施形態のコンピュータシステム 1 0 0 においては、バスライン 2 2 0 を介して、中央演算処理装置（CPU）2 2 1，RAM 2 2 2，ハードディスクコントローラ 2 2 3，フロッピーディスクドライバ 2 2 4，CD-ROM ドライバ 2 2 5，マウスコントローラ 2 2 6，キーボードコントローラ 2 2 7，ディスプレイコントローラ 2 2 8，モデム 2 2 9 および ROM 2 3 0 が相互に接続されている。

#### 【 0 0 2 6 】

そして、フロッピーディスクドライバ 2 2 4 および CD-ROM ドライバ 2 2 5 は、図 3 を参照して説明したように、それぞれ、フロッピーディスク 2 1 2 および CD-ROM 2 1 0 が装填され、装填されたフロッピーディスク 2 1 2 および CD-ROM 2 1 0 に対するアクセスを行なうものである。

また、ハードディスクコントローラ 2 2 3 は、ハードディスク 2 1 1 を制御するとともに CPU 2 2 1 からの指示に従ってハードディスク 2 1 1 に対するアクセスを行なうものであり、マウスコントローラ 2 2 6 は、マウス 1 0 4 とコンピュータシステム 1 0 0 との間のインタフェースとして機能するものであり、キーボードコントローラ 2 2 7 は、キーボード 1 0 3 とコンピュータシステム 1 0 0 との間のインタフェースとして機能するものであり、ディスプレイコントローラ 2 2 8 は、CPU 2 2 1 からの制御信号に応じて CRT ディスプレイ 1 0 2 の表示状態を制御するものである。

#### 【 0 0 2 7 】

さらに、モデム 2 2 9 は、図示しない通信回線に接続され、図 3 および図 4 に示したものと同様構成の、他のコンピュータシステムとの通信を行なうためのも

のである。

前述したように、CD-ROM 210 には支援プログラム 300 (図 2 参照) が記憶されており、CD-ROM ドライバ 225 により、その CD-ROM 210 からその支援プログラム 300 が読み込まれ、バスライン 220 を経由し、ハードディスクコントローラ 223 によりハードディスク 211 内に格納される。実際の実行に当たっては、ハードディスク 211 内のプログラム 300 は、RAM 222 上にロードされ、CPU 221 により実行される。

【0028】

支援プログラム 300 は、図 1 に示す組込み支援システム 10 を実現するためのものであり、CPU 10 が、バスライン 220 を介して RAM 222 におけるプログラム 300 を実行することにより、支援システム 10 を成すメカ設計部 20、3 次元リアルタイムシミュレーション装置 30、状態遷移図・表作成編集装置 41、統合開発環境 43 およびインタフェース部 50～52 としての機能（その詳細については後述）が実現され、本実施形態の支援システム 10 が実現されるようになっている。

【0029】

この支援プログラム 300 は、前述のごとく CD-ROM 210 に記録された形態で提供されるほか、その他、フレキシブルディスク（フロッピーディスク 212）等の、コンピュータ読取可能な記録媒体に記録された形態で提供される。そして、コンピュータシステム 100 はその記録媒体からプログラム 300 を読み取って、前述したようにハードディスク 211 等の内部記憶装置または外部記憶装置に転送し格納して用いる。また、そのプログラム 300 を、例えば磁気ディスク、光ディスク、光磁気ディスク等の記憶装置（記録媒体）に記録しておき、その記憶装置から通信回線およびモデム 229 を介してコンピュータシステム 100 に提供するようにしてもよい。

【0030】

なお、本実施形態における記録媒体としては、上述したフレキシブルディスク、CD-ROM、磁気ディスク、光ディスク、光磁気ディスクのほか、IC カード、ROM カートリッジ、磁気テープ、パンチカード、コンピュータの内部記憶

装置（RAMやROMなどのメモリ）、外部記憶装置等や、バーコードなどの符号が印刷された印刷物等の、コンピュータ読取可能な種々の媒体を利用することができる。

#### 【0031】

さて、図4に示すコンピュータシステム100により実現される、本実施形態の支援システム10は、アクチュエータやセンサを有し3次元的な動作を行なう機構の設計・開発に際し、機構を制御すべくその機構に組み込まれる制御プログラム（組込みソフトウェア）の開発を支援するためのものである。

そして、上述のような機能を実現すべく、本実施形態の支援システム10は、図1に示すように、メカ設計部20、3次元リアルタイムシミュレーション装置30、状態遷移図・表作成編集装置41、統合開発環境装置43およびインタフェース部50～52としての機能を有するほか、開発された組込みソフトウェアを実際に格納されて機構制御を行なうマイクロコンピュータチップ（以下、マイコンチップという）42も有している。

#### 【0032】

なお、図1は、本発明の一実施形態としての支援システムの機能的構成を示すブロック図である。

本実施形態のシステム10は、メカ設計と組込みソフトウェア開発とをコンカレントに遂行するためのもので、大きく分けると、メカ設計部20、3次元リアルタイムシミュレーション装置30および組込みソフトウェア開発部40から構成されている。

#### 【0033】

ここで、メカ設計部（意匠設計・3次元CADシステム、機構設計部）20は、アクチュエータおよびセンサを含む複数の部品からなる機構を3次元的に設計するものである。このメカ設計部20と3次元リアルタイムシミュレーション装置30との連携の詳細については、図5および図6を参照しながら後述する。

3次元リアルタイムシミュレーション装置（3次元機構モデルシミュレーション部）30は、メカ設計部20により設計された機構を3次元機構モデルとして内部に構築され、その機構の動作をシミュレートするものである。この3次元リ

アルタイムシミュレーション装置 3 0 の詳細については、図 1 7 ～ 図 8 2 を参照しながら後述する。

【 0 0 3 4 】

組込みソフトウェア開発部 4 0 は、メカ設計部 2 0 による機構の設計と並行してその機構に組み込まれその機構の動作を制御するための制御プログラムを組込みソフトウェアとして開発するものである。

また、本実施形態のシステム 1 0 においては、インタフェース部 5 0 ～ 5 2 がそなえられるほか、組込みソフトウェア開発部 4 0 は、状態遷移図・表作成編集装置 4 1，マイコンチップ 4 2 および統合開発環境装置 4 3 をそなえて構成されている。

【 0 0 3 5 】

インタフェース部（第 1 インタフェース部） 5 0 は、メカ設計部 2 0 による設計データを、3 次元リアルタイムシミュレーション装置 3 0 にて構築される 3 次元機構モデルに動的に反映させるべく、メカ設計部 2 0 から 3 次元リアルタイムシミュレーション装置 3 0 へ入力する機能を果たすとともに、3 次元リアルタイムシミュレーション装置 3 0 によるシミュレーション結果を、機構の設計に動的に反映させるべく 3 次元リアルタイムシミュレーション装置 3 0 からメカ設計部 2 0 へ入力する機能も果たすように構成されている。

【 0 0 3 6 】

このインタフェース部 5 0 の機能は、メカ設計部 2 0 と 3 次元リアルタイムシミュレーション装置 3 0 とのいずれか一方に含まれてもよいし、これらのメカ設計部 2 0 と 3 次元リアルタイムシミュレーション装置 3 0 とに分割されて含まれていてもよい。

インタフェース部（第 2 インタフェース部） 5 1，5 2 は、いずれも、3 次元リアルタイムシミュレーション装置 3 0 の動作と組込みソフトウェア開発部 4 0 の動作との同期をとりながら、3 次元リアルタイムシミュレーション装置 3 0 と組込みソフトウェア開発部 4 0 との間でデータ送受を行なうものである。

【 0 0 3 7 】

また、本実施形態のシステム 1 0 における組込みソフトウェア開発部 4 0 は、

状態遷移図・表作成編集装置（状態遷移図・表作成部）4 1，マイコンチップ 4 2 および統合開発環境装置 4 3 を含んで構成されている。

ここで、状態遷移図・表作成編集装置 4 1 は、組込みソフトウェアの詳細設計を行なうべく組込みソフトウェアの仕様を記述し、組込みソフトウェアについてのタスク制御フローを作成するものである。この状態遷移図・表作成編集装置 4 1 の詳細、および、状態遷移図・表作成編集装置 4 1 と 3 次元リアルタイムシミュレーション装置 3 0 との連携の詳細については、図 7 ～図 1 4 を参照しながら後述する。

#### 【 0 0 3 8 】

マイコンチップ 4 2 は、開発中の組込みソフトウェアを組み込まれ、実際に機構に搭載されてその機構の動作を制御するものであり、統合開発環境装置 4 3 は、状態遷移図・表作成編集装置 4 1 からのタスク制御フローを、C，C++，アセンブラコードに変換し（コーディング，コードジェネレーション）、ターゲットマイコン用の実行モジュール（即ち、マイコンチップ 4 2 に書き込まれるべき組込みソフトウェア）を作成するためのものである。

#### 【 0 0 3 9 】

そして、インタフェース部 5 1 は、3 次元リアルタイムシミュレーション装置 3 0 の動作と状態遷移図・表作成編集装置 4 1 の動作との同期をとりながら、3 次元リアルタイムシミュレーション装置 3 0 と状態遷移図・表作成編集装置 4 1 との間でデータ送受を行なうものである。

本実施形態では、図 1 0 ～図 1 4 を参照しながら後述するごとく、状態遷移図・表作成編集装置 4 1 が、複数のタスクを並行して実行するマルチタスクを採用し、3 次元リアルタイムシミュレーション装置 3 0 のシミュレーション動作期間中、複数のタスクを停止させるように機能する同期タスクを複数のタスクとは別に実行し、インタフェース部 5 1 が、同期タスクを用いて 3 次元リアルタイムシミュレーション装置 3 0 の動作と状態遷移図・表作成編集装置 4 1 の動作とを同期させている。

#### 【 0 0 4 0 】

このとき、インタフェース部 5 1 は、同期タスクの優先度を最も高く設定し、

この同期タスクにより複数のタスクの開始・停止を制御して、3次元リアルタイムシミュレーション装置30の動作と状態遷移図・表作成編集装置41の動作とを同期させている。

このインタフェース部51の機能は、3次元リアルタイムシミュレーション装置30と状態遷移図・表作成編集装置41とのいずれか一方に含まれてもよいし、これらの3次元リアルタイムシミュレーション装置30と状態遷移図・表作成編集装置41とに分割されて含まれていてもよい。

#### 【0041】

また、インタフェース部52は、3次元リアルタイムシミュレーション装置30の動作とマイコンチップ42の動作との同期をとりながら、3次元リアルタイムシミュレーション装置30とマイコンチップ42との間でデータ送受を行なうものである。

本実施形態では、マイコンチップ42も、状態遷移図・表作成編集装置41と同様、複数のタスクを並行して実行するマルチタスクを採用し、3次元リアルタイムシミュレーション装置30のシミュレーション動作期間中、該複数のタスクを停止させるように機能する同期タスクを複数のタスクとは別に実行し、インタフェース部52が、同期タスクを用いて3次元リアルタイムシミュレーション装置30の動作とマイコンチップ42の動作とを同期させている。

#### 【0042】

このとき、インタフェース部52も、インタフェース部51と同様、同期タスクの優先度を最も高く設定し、この同期タスクにより複数のタスクの開始・停止を制御して、3次元リアルタイムシミュレーション装置30の動作とマイコンチップ42の動作とを同期させている。

このインタフェース部52の機能は、3次元リアルタイムシミュレーション装置30とマイコンチップ42とのいずれか一方に含まれてもよいし、これらの3次元リアルタイムシミュレーション装置30とマイコンチップ42とに分割されて含まれていてもよい。

#### 【0043】

なお、本実施形態のシステム10では、インタフェース部51、52が、それ

ぞれ、状態遷移図・表作成編集装置 4 1 およびマイコンチップ 4 2 から 3 次元リアルタイムシミュレーション装置 3 0 へ、3 次元機構モデルにおけるアクチュエータに対するアクチュエータ指令信号を受け渡すとともに、3 次元リアルタイムシミュレーション装置 3 0 から状態遷移図・表作成編集装置 4 1 およびマイコンチップ 4 2 へ、そのアクチュエータ指令信号に応じたシミュレーション結果として得られる、3 次元機構モデルにおけるセンサからのセンサ信号を受け渡すように構成されている。

【 0 0 4 4 】

また、3 次元リアルタイムシミュレーション装置 3 0 において、ロジックアナライザ（解析部；図 1 5 参照）をそなえてもよい。このロジックアナライザは、アクチュエータに対するアクチュエータ指令信号と 3 次元リアルタイムシミュレーション装置 3 0 で得られたセンサ信号との時間変化をリアルタイムで解析して表示し、組込みソフトウェア開発者がその時間変化を確認できるようにするためのものである。

【 0 0 4 5 】

さらに、新たに、2 次元操作パネルを構築する手段と 3 次元リアルタイムシミュレーション装置 3 0 とを結び付けるインタフェース手段を有し、3 次元機構モデルの操作を上記 2 次元操作パネル上からインタラクティブにできるように構成してもよい。この場合、2 次元操作パネル上には、実際にマイコンチップ 4 2 上にそなえられるスイッチ類のイメージが表示されている。そして、組込みソフトウェア開発者は、2 次元操作パネル上のスイッチ類のイメージをマウス等により操作することで、マイコンチップ 4 2 上のスイッチ類を操作した場合のシミュレーションを、3 次元リアルタイムシミュレーション装置 3 0 に実行させることができるようになっている。

【 0 0 4 6 】

またさらに、本実施形態の 3 次元リアルタイムシミュレーション装置 3 0 は、ギア、カム、溝、クラッチ、ローラ、ラチェット等の各種機構についての機構モデルを作成する手段を有するとともに、DC モータ、ステッピングモータ、AC モータ、マイクロスイッチ／光学スイッチ、エンコーダ、ポテンシオメータ等の

各種モータ／センサについての機構モデルを作成する手段を有しており、これらの手段を用いて、各種機構や各種モータ／センサの任意の組み合わせからなる機構を3次元機構モデルとして表現することができるようになっている。

## 【0047】

一方、図2は、本実施形態のコンピュータ読み取り可能な記録媒体300に記録された、支援プログラム310の構成を示す図である。

この図2に示す記録媒体300は、支援プログラム310を記録されたCD-ROM210や、そのプログラム310をインストールされた状態のハードディスク211や、そのプログラム310をダウンロードされたフロッピーディスク212、RAM222あるいはROM230等を代表的に示したものである。また、記録媒体300は、通信回線等で接続された複数の記録媒体であって、これらの記録媒体に、プログラム310を成す複数のプログラム311～316を分割して記録したものも含んでいる。

## 【0048】

このような記録媒体300に記録された、本実施形態の支援プログラム310は、アクチュエータおよびセンサを含む複数の部品からなる機構を制御すべくその機構に制御プログラムとして組み込まれる組込みソフトウェアの開発を支援する機能（つまり図1に示すシステム10）を、図3や図4に示すようなコンピュータシステム100により実現するためのもので、機構設計プログラム311、3次元機構モデルシミュレーションプログラム312、組込みソフトウェア開発プログラム313、第1インタフェースプログラム314、第2インタフェースプログラム315および統合開発環境プログラム316を含んで構成されている。

## 【0049】

ここで、機構設計プログラム311は、前述したメカ設計部20としてコンピュータシステム100を機能させるものであり、3次元機構モデルシミュレーションプログラム312は、前述した3次元リアルタイムシミュレーション装置30としてコンピュータシステム100を機能させるものである。

組込みソフトウェア開発プログラム313は、前述した組込みソフトウェア開



発部 4 0 としてコンピュータシステム 1 0 0 を機能させるもので、前述した状態遷移図・表編集装置 4 1 としてコンピュータシステム 1 0 0 を機能させる状態遷移図・表作成プログラム 3 2 0 を含んで構成されている。

【 0 0 5 0 】

第 1 インタフェースプログラム 3 1 4 は、前述したインタフェース部 5 0 としてコンピュータシステム 1 0 0 を機能させるものである。この第 1 インタフェースプログラム 3 1 4 は、機構設計プログラム 3 1 1 と 3 次元機構モデルシミュレーションプログラム 3 1 2 とのいずれか一方に含まれてもよいし、これらのプログラム 3 1 1 および 3 1 2 に分割されて含まれていてもよい。

【 0 0 5 1 】

なお、機構設計プログラム 3 1 1 (メカ設計部 2 0) は、コンピュータシステム 1 0 0 内にローディングされた第 1 インタフェースプログラム 3 1 4 (インタフェース部 5 0) と結合され、このコンピュータシステム 1 0 0 内において、3 次元機構モデルシミュレーションプログラム 3 1 2 の動作つまり 3 次元機構モデルの動作に連携して実行される。

【 0 0 5 2 】

第 2 インタフェースプログラム 3 1 5 は、前述したインタフェース部 5 1 および 5 2 としてコンピュータシステム 1 0 0 を機能させるものである。

この第 2 インタフェースプログラム 3 1 5 のうちインタフェース部 5.1 にかかる部分については、3 次元機構モデルシミュレーションプログラム 3 1 2 と状態遷移図・表作成プログラム 3 2 0 とのいずれか一方に含まれてもよいし、これらのプログラム 3 1 2 および 3 2 0 に分割されて含まれていてもよい。

【 0 0 5 3 】

また、第 2 インタフェースプログラム 3 1 5 のうちインタフェース部 5 2 にかかる部分については、3 次元機構モデルシミュレーションプログラム 3 1 2 とマイコンチップ 4 2 に格納される制御プログラム (組込みソフトウェア) とのいずれか一方に含まれてもよいし、これらのプログラム 3 1 2 および制御プログラムに分割されて含まれていてもよい。

【 0 0 5 4 】

統合開発環境プログラム 3 1 6 は、前述した統合開発環境装置 4 3 としてコンピュータシステム 1 0 0 を機能させるものである。

なお、本実施形態では、前述した通り、状態遷移図・表作成プログラム 3 2 0 (状態遷移図・表作成編集装置 4 1) やマイコンチップ 4 2 がマルチタスクを採用している。

【0 0 5 5】

そのため、第 2 インタフェースプログラム 3 1 5 は、状態遷移図・表作成プログラム 3 2 0 またはマイコンチップ 4 2 で実行される複数のタスクを、優先度を最も高く設定した、一つの同期タスクによって代表させ、その同期タスクを用いて、3 次元機構モデルシミュレーションプログラム 3 1 2 (3 次元リアルタイムシミュレーション装置 3 0) がシミュレーション動作を行なっている間は、状態遷移図・表作成プログラム 3 2 0 またはマイコンチップ 4 2 での複数のタスクを停止させ、これにより、3 次元機構モデルシミュレーションプログラム 3 1 2 による動作と、状態遷移図・表作成プログラム 3 2 0 やマイコンチップ 4 2 による動作とを同期させている。

【0 0 5 6】

このとき、第 2 インタフェースプログラム 3 1 5 (インタフェース部 5 1, 5 2) は、コンピュータシステム 1 0 0 で実行される際、3 次元機構モデルシミュレーションプログラム 3 1 2 と状態遷移図・表作成プログラム 3 2 0 またはマイコンチップ 4 2 (制御プログラム) との間において、以下のような通信を行なうことにより、同期タスクを用いた同期化を行なっている。

【0 0 5 7】

即ち、3 次元機構モデルシミュレーションプログラム 3 1 2 が、状態遷移図・表作成プログラム 3 2 0 または制御プログラムの実行時間 ( $T_S$ ) を第 2 インタフェースプログラム 3 1 5 に通知する。

第 2 インタフェースプログラム 3 1 5 (インタフェース部 5 1, 5 2) は、その通知を受けて、通知された実行時間分だけ状態遷移図・表作成プログラム 3 2 0 または制御プログラムの実行を許容し、状態遷移図・表作成プログラム 3 2 0 または制御プログラムがその実行時間分だけ実行されると、状態遷移図・表作成

プログラム 3 2 0 または制御プログラムの実行をその段階で中断させ、その実行時間分だけのプログラム実行が終了したと、そのプログラム実行により得られたアクチュエータ指令信号とを 3 次元機構モデルシミュレーションプログラム 3 1 2 に通知する。

【 0 0 5 8 】

この終了通知を受けた 3 次元機構モデルシミュレーションプログラム 3 1 2 は、3 次元機構モデルを、状態遷移図・表作成プログラム 3 2 0 または制御プログラムの、その実行時間分の制御に応じた動作量だけ動作させ、その動作量に応じたセンサ信号を得てから、状態遷移図・表作成プログラム 3 2 0 または制御プログラムの次の実行時間を決定し、その決定した実行時間と前記センサ信号とを第 2 インタフェースプログラム 3 1 5 に通知する。

【 0 0 5 9 】

このような通信処理を繰り返すことにより、相互に同期がとられた状態で、3 次元機構モデルシミュレーションプログラム 3 1 2 と状態遷移図・表作成プログラム 3 2 0 または制御プログラムとが実行され、3 次元リアルタイムシミュレーション装置 3 0 においては、その状態遷移図・表作成プログラム 3 2 0 または制御プログラムの実行による制御に従って 3 次元機構モデルが動作する。

【 0 0 6 0 】

また、コンピュータシステム 1 0 0 (リアルタイムシミュレーション装置 3 0 ) 内に構築された 3 次元機構モデルは、CD-ROM 2 1 0 からこのコンピュータシステム 1 0 0 内にローディングされた支援プログラム 3 1 0 を構成する 3 次元機構モデルシミュレーションプログラム 3 1 2 によりそのコンピュータシステム 1 0 0 内で動作する。

【 0 0 6 1 】

そして、コンピュータシステム 1 0 0 内に組み込まれた状態遷移図・表作成プログラム 3 2 0 や、マイコンチップ 4 2 に組み込まれた制御プログラム (組込みソフトウェア) は、コンピュータシステム 1 0 0 内にローディングされた第 2 インタフェースプログラム 3 1 5 (インタフェース部 5 1, 5 2) と結合され、このコンピュータシステム 1 0 0 内において、3 次元機構モデルシミュレーション

プログラム 3 1 2 の動作つまり 3 次元機構モデルの動作と同期して実行される。

【 0 0 6 2 】

上述のごとく、本実施形態の支援システム 1 0 は、1 つの装置（ここではコンピュータシステム 1 0 0）内に、メカ設計部 2 0，3 次元リアルタイムシミュレーション装置 3 0，状態遷移図・表作成編集装置 4 1，統合開発環境装置 4 3 およびインタフェース部 5 0 ～ 5 2 の全てを組み込んだ形態となっている。

本発明の支援システムは、上述のような形態以外に、複数のコンピュータシステム 1 0 0 をモデム 2 2 9 および通信回線（図示省略）により相互に通信可能に接続し、これらのコンピュータシステム 1 0 0 に、メカ設計部 2 0，3 次元リアルタイムシミュレーション装置 3 0，状態遷移図・表作成編集装置 4 1，統合開発環境装置 4 3 およびインタフェース部 5 0 ～ 5 2 としての機能を分担させた形態として構成することもできる。

【 0 0 6 3 】

この場合、機構設計プログラム 3 1 1，3 次元機構モデルシミュレーションプログラム 3 1 2，状態遷移図・表作成プログラム 3 2 0，第 1 インタフェースプログラム 3 1 4，第 2 インタフェースプログラム 3 1 5 および統合開発環境プログラム 3 1 6 は、それぞれ、メカ設計部 2 0，3 次元リアルタイムシミュレーション装置 3 0，状態遷移図・表作成装置 4 1，インタフェース部 5 0 ～ 5 2 および統合開発環境装置 4 3 としての機能を果たすコンピュータシステム 1 0 0 にインストールされることになる。

【 0 0 6 4 】

なお、本実施形態のシステム 1 0 において、開発中の制御プログラム（組込みソフトウェア）をローディングしたマイコンチップ 4 2 は、コンピュータシステム 1 0 0 に I / O ボード（図示省略）を介して接続されており、制御プログラムに基づいてマイコンチップ 4 2 から出力されたアクチュエータ指令信号（本来はモータドライバ等に伝達される電気信号）は、前記 I / O ボードを介してコンピュータシステム 1 0 0 に入力され、インタフェース部 5 2 を介して 3 次元リアルタイムシミュレーション装置 3 0 に入力される。そして、3 次元リアルタイムシミュレーション装置 3 0 は、そのアクチュエータ指令信号に応じて 3 次元機構モ

デルを動作させ、本来センサから出力される信号と同様のセンサ信号を生成し、そのセンサ信号を、インタフェース部 5 2 および前記 I / O ボードを介してマイコンチップ 4 2 に入力している。

【 0 0 6 5 】

このようにして、本実施形態の支援システム 1 0 では、3 次元リアルタイムシミュレーション装置 3 0 を中核に据えて、メカ試作品を作らなくても組込みソフトウェア（制御プログラム）の開発をメカ設計とは別個（コンカレント）に進めることができるようになっているほか、状態遷移図・表によるタスク制御記述法と 3 次元のリアルタイムシミュレーション装置 3 0 とを組み合わせ、組込みソフトウェアをメカ試作品なしに効率よく開発することができるようになっている。

【 0 0 6 6 】

上述のごとく、本実施形態の支援システム 1 0 は、リアルタイムに機構シミュレーションを行ないながらメカ設計を行なうメカ設計部 2 0 と、機構をリアルタイムにシミュレーションする 3 次元リアルタイム・シミュレーション装置 3 0 と、3 次元リアルタイムシミュレーション装置 3 0 上の仮想的な 3 次元機構モデルを実際のメカに見立てて組込みソフトウェアを開発する組込みソフトウェア開発部 4 0 とをそなえ、これらをインタフェース部 5 0 ～ 5 2 を介して連携させるように構成されている。

【 0 0 6 7 】

このような本実施形態のシステム 1 0 においては、メカ設計とそのメカを制御するための組込みソフトウェアの開発とがコンカレントに実行される。次に、その基本的な実行手順について簡単に説明する。

メカ設計者は、メカ設計部（意匠設計・CG システム、3 次元 CAD システム）2 0 と 3 次元リアルタイムシミュレーション装置 3 0 とを、インタフェース部 5 0 を介して連動させ、関節、ギア、カム等の各種機構の動きを詳細に検証しながら、単体部品の設計、および、それらを組み立てるためのアセンブリ設計を進めていく。この設計過程で、メカ設計者は、設計変更が生じる度に、その変更部分についての設計データを、インタフェース部 5 0 を介して 3 次元リアルタイムシミュレーション装置 3 0 に動的に反映し、その時の全体アセンブリの動きを確

かめる。3次元リアルタイムシミュレーション装置30で扱う3次元機構モデルはソフトウェアシステムであるため、実際に試作品を作製する場合と違って任意の設計変更を速やかに反映することが可能である。

【0068】

一方、組込みソフトウェア開発者は、3次元リアルタイムシミュレーション装置30を実機と見立て、この3次元リアルタイムシミュレーション装置30と状態遷移図・表作成編集装置41およびマイコンチップ42とを、インタフェース部51および52を介して連動させながら組込みソフトウェア開発を進める。このとき、メカ設計側で発生した任意の設計変更は、3次元リアルタイムシミュレーション装置30に速やかに反映されるため、組込みソフトウェアの開発がメカ側の設計変更によって滞ることはない。

【0069】

組込みソフトウェア開発者は、まず、状態遷移図・表編集装置41と3次元リアルタイムシミュレーション装置30とを連動させながら組込みソフトウェアのタスク制御の詳細仕様（タスク制御フロー）を決定し、その結果を統合開発環境装置（マイコンシステム開発環境）43に渡す。

統合開発環境装置43では、状態遷移図・表作成編集装置41から受け取ったタスク制御フローがC、C++、アセンブラコードに変換され、ターゲットマイコン用の実行モジュール（組込みソフトウェア）が作成されてマイコンチップ42に組み込まれる。

【0070】

マイコンチップ42と3次元リアルタイムシミュレーション装置30とは、インタフェース部52を介して連動することができるため、組込みソフトウェア開発部40では、マイコンチップ42に組み込んだ実行モジュールが仕様通りのタスク制御を行なうか否かを、3次元リアルタイムシミュレーション装置30上の仮想メカモデル（3次元機構モデル）を動かすことにより確かめることができる。

【0071】

このように、本実施形態のシステム10では、3次元リアルタイムシミュレー

ション装置 3 0 を仲介としてメカ設計および組込みソフトウェア開発をコンカレントに進め、最終的には、マイコンチップ 4 2 と実機とを接続して、実機を用いた最終結合テストを行なうことで、極めて効率よく、メカ設計および組込みソフトウェア開発が可能になる。

#### 【 0 0 7 2 】

次に、図 5 ～図 8 2 を参照しながら、本実施形態の支援システム 1 0 の各部の詳細について説明する。

まず、メカ設計部（意匠設計・3次元CADシステム）2 0 と3次元リアルタイムシミュレーション装置 3 0 との連携部分（インタフェース部 5 0）について、図 5 および図 6 を参照しながら説明する。

#### 【 0 0 7 3 】

なお、図 5 はシステム 1 0 の要部（メカ設計部 2 0 および3次元リアルタイムシミュレーション装置 3 0）を示すブロック図、図 6（A）および図 6（B）は、それぞれ、メカ設計部 2 0 から3次元リアルタイムシミュレーション装置 3 0 へ受け渡される図形データファイルおよびアセンブリデータファイルの具体例を示す図である。

#### 【 0 0 7 4 】

図 5 に示すように、メカ設計部（意匠設計・3次元CADシステム）2 0 から、設計対象の各部品要素の図形データファイル（.slpファイルフォーマット）と、各部品要素を組み立てた状態にしたアセンブリデータファイル（.asyファイルフォーマット）とが出力され、これらのデータファイルは、3次元リアルタイムシミュレーション装置 3 0 に読み込まれる。

#### 【 0 0 7 5 】

このとき、メカ設計部 2 0 と3次元リアルタイムシミュレーション装置 3 0 との間には、前述したインタフェース部 5 0（図 1 参照）が介在しているが、図 5 に示す例では、インタフェース部 5 0 としての機能が、メカ設計部 2 0 と3次元リアルタイムシミュレーション装置 3 0 とのいずれか一方に含まれているか、もしくは、これらのメカ設計部 2 0 と3次元リアルタイムシミュレーション装置 3 0 とに分割されて含まれている。従って、図 5 では、インタフェース部 5 0 は図

示されていない。

【0076】

図6（A）に示すように、図形データファイルは.slpのファイルフォーマットであり、この.slpファイルフォーマットは、三角形ポリゴンの集合体を記述したもので、各三角形ポリゴンの各頂点の法線と3次元位置座標とを順次書き出したものである。また、図6（B）に示すように、アセンブリデータファイルは.asyのファイルフォーマットであり、この.asyファイルフォーマットでは、各部品要素の相互の配置情報が定義されている。

【0077】

3次元リアルタイムシミュレーション装置30は、メカ設計部20から、.asyおよび.slpのファイルを読み込んだ後、関節、カム、ギア等の機構情報やモータ、センサの各特性情報を、3次元リアルタイムシミュレーション装置20のGUI（Graphical User Interface）を通じて入力し、仮想メカモデル（3次元機構モデル）を完成させる。

【0078】

次に、3次元リアルタイムシミュレーション装置30と状態遷移図・表作成編集装置41との連携部分（インタフェース部51）について、図7～図14を参照しながら説明する。

なお、図7はシステム10の要部（3次元リアルタイムシミュレーション装置30、状態遷移図・表作成編集装置41およびインタフェース部51）を示すブロック図、図8は状態遷移図の表示例を示す図、図9は状態遷移表の表示例を示す図、図10（A）～図10（C）はマルチタスクについて説明するためのタイムチャート、図11（A）および図11（B）は優先度によるタスクのスケジューリングについて説明するための図、図12（A）～図12（D）は同期タスクを用いた同期化処理について説明するためのタイムチャートである。

【0079】

図7に示すように、状態遷移図・表作成編集装置41には、状態遷移で記述したタスク制御を実行するタスク制御部411と、キーボードやマウス等を使ってタスク制御部411によるタスク制御を編集するユーザインタフェース部412



と、タスク制御部411によるタスク制御の内容を表示する状態遷移表示部413と、3次元リアルタイムシミュレーション装置30とのインタフェースを取るインタフェース部414とがそなえられている。状態遷移表示部413では、タスク制御部411によるタスク制御の内容（タスク制御フロー）が、図8に示すようにグラフィカルに状態遷移図の形で、または、図9に示すようにテーブル形式（状態遷移表の形）で表示されるようになっている。

#### 【0080】

また、3次元リアルタイム・シミュレーション装置30には、リアルタイムに3次元機構モデルのシミュレーションを行なうシミュレーション部31と、状態遷移図・表作成編集装置41とのインタフェースを取るインタフェース部32とがそなえられている。

インタフェース部414とインタフェース部32とが、上述したインタフェース部51を構成している。つまり、図7に示す例では、インタフェース部51としての機能が、3次元リアルタイムシミュレーション装置30と状態遷移図・表作成編集装置41とに分割されて含まれている。

#### 【0081】

ところで、ファームウェア（状態遷移図・表作成編集部41やマイコンチップ42）と連携した3次元リアルタイムシミュレーション装置30では、ファームウェア上での実時間と同じ時間スケールで、3次元機構モデル（仮想メカモデル）の動きなどをシミュレートする必要がある。

例えば図10（A）および図10（B）に示すようにファームウェアのタスク1，2で $T_S$ 時間の計算処理が実行された場合には、引き続き、図10（C）に示すように、3次元リアルタイムシミュレーション装置30で、仮想メカ（3次元機構モデル）について $T_S$ 時間分のシミュレーションを実行する。

#### 【0082】

ただし、図10（A）および図10（B）に示すように、複数（図中、2つ）のタスクが並列に走るとき、複数のタスク間で同期をとりながら同時に処理の停止／開始を行なうことは、どのタスクが時間の管理を行なうか、どのように処理を中断／再開するかなどの問題がある。

そこで、本実施形態では、図 1 2 (C) に示すような同期用のタスク（同期タスク）を用意し、この同期タスクに時間の管理と他のタスクの開始／停止の制御を行なわせることで、タスク内に特別な処理を追加せずに、ファームウェアと 3 次元リアルタイムシミュレーション装置 3 0 とを同期させている。

#### 【0083】

このとき、リアルタイム OS の基本機能である、優先度によるタスクのスケジューリング機能を用いる。複雑な組込みソフトウェア（ファームウェア）では、リアルタイム OS を用いて、複数のタスクにより様々な処理を行なっている。リアルタイム OS では、複数のタスクを並列に走らせることができるが、リソース（CPU）は 1 つしかないので、OS が、どのタスクに CPU を割り当てるかスケジューリングを行なっている。

#### 【0084】

このスケジューリングを行なうべく、タスクに優先度を設定し、複数のタスクから CPU 使用の要求が同時にきたとき、優先度の高いタスクから CPU を割り当てていく手法が一般的に採用されている。例えば図 1 1 (A) および図 1 1 (B) に示すように、優先度の低いタスク L の処理中に優先度の高いタスク H が処理を開始する時には、OS は、タスク L の処理を途中で停止させ、処理の実行権をタスク H に移す。そして、OS は、タスク H の処理が終了すると、タスク L の中断していた処理を再開させる。

#### 【0085】

このような優先度によるタスク切り替え機能を用い、同期タスクの優先度を他のタスクよりも高く設定すれば、同期タスクが他のタスクを開始／停止する制御を行なうことができる。このとき、通常のタスクには同期処理用の特別な処理は何も必要ない。

例えば図 1 2 (A) ～図 1 2 (D) に示すように、2 つのタスク 1, 2 を並列に実行する場合、新たに追加した同期タスクの優先度を最も高く設定し、タスク 1, 2 での  $T_s$  時間の計算処理に対応して 3 次元リアルタイムシミュレーション装置 3 0 で  $T_s$  時間分のシミュレーションを実行している期間中は、CPU による処理の実行権を同期タスクに移す。これにより、その期間中、タスク 1, 2 の

処理が停止され、 $T_S$ 時間分のシミュレーション終了後にタスク 1, 2 の処理が再開される。

【 0 0 8 6 】

図 1 3 は、本実施形態における同期タスクを用いた同期化処理について、より詳細に説明するためのブロック図であり、この図 1 3 に示す例では、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）と 3 次元リアルタイムシミュレーション装置 3 0 とのインタフェース部 5 1（5 2）を共有メモリ 5 3 により実現している。

【 0 0 8 7 】

図 1 3 に示す状態遷移図・表作成編集装置 4 1 では、 $n$  個のタスク 1 ～  $n$  が、それぞれ内部メモリ 4 1 5 のデータを参照しながら並列的に実行される。同期タスクの処理中は、内部メモリ 4 1 5 のデータ（アクチュエータ指令信号）を読み出して 3 次元リアルタイムシミュレーション装置 3 0 に送信し、この 3 次元リアルタイムシミュレーション装置 3 0 からのデータ（センサ信号）を受信して内部メモリ 4 1 5 に書き込む。

【 0 0 8 8 】

このとき、各タスクの処理に合わせて個々のデータを 3 次元リアルタイムシミュレーション装置 3 0（仮想メカ）と送受信してもよいが、本実施形態では、通信を効率的に行なうために、内部メモリ 4 1 5 にバッファリングしておき、同期タスクが、必要なときにまとめてデータの送受信を行なう。通信手法としては、上述のような共有メモリ 5 3 を用いる手法のほかに、Socket 通信やメッセージ通信などによる方法を用いることもできる。

【 0 0 8 9 】

次に、同期タスクを用いた同期化処理について、図 1 4 に示すフローチャート（ステップ S 1 1 ～ S 1 9）従って説明する。

この図 1 4 に示すように、3 次元リアルタイムシミュレーション装置 3 0 からは前回のシミュレーション結果（センサ信号）とともに次回の計算時間（処理時間） $T_S$  が送信され、その時間  $T_S$  が状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）に設定される（ステップ S 1 1）。

## 【 0 0 9 0 】

そして、その時間  $T_S$  だけ同期タスクの処理が停止される（ステップ S 1 2）。つまり、その時間  $T_S$  の間、複数のタスクの処理が実行され、各タスクの処理によって得られたアクチュエータ指令信号（モータ指令信号）が内部メモリ 4 1 5 に書き込まれる。本実施形態では、同期タスクは、通常は停止していて、ある時間  $T_S$  が経過すると起動されるようにする。これは、OS が持っているタイマ機能やスリープ機能により実現することができる。

## 【 0 0 9 1 】

時間  $T_S$  だけ同期タスクの処理を停止して待機した後（ステップ S 1 2）、同期タスクを起動して他の全てのタスクを停止し、同期タスクで、各タスクからの出力データ（指令信号）を内部メモリ 4 1 5 から読み込み（ステップ S 1 3）、その出力データを共有メモリ 5 3 に書き込んでから（ステップ S 1 4）、共有メモリ 5 3 に同期フラグ 1（Loop Flag）として“0”を書き込み（ステップ S 1 5）、3次元リアルタイムシミュレーション装置 3 0 に処理の実行を許可する。これに伴い、共有メモリ 5 3 において同期フラグ 2（Ready Flag）が“0”から“1”に設定される。

## 【 0 0 9 2 】

この後、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）では、3次元リアルタイムシミュレーション装置 3 0 からのシミュレーション終了通知を待つとともに、他のタスクを停止したままにしておく。

3次元リアルタイムシミュレーション装置 3 0 は、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）からシミュレーション実行開始の指令を受け取ったら、 $T_S$  時間分のシミュレーションを行ない、 $T_S$  時間分のシミュレーションを完了すると、シミュレーション結果（センサ信号）と次回の計算時間（処理時間） $T_S$  とを共有メモリ 5 3 に書き込むとともに、共有メモリ 5 3 上の同期フラグ 2（Ready Flag）を“0”に書き換え、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）から次のシミュレーション実行開始の指令を受けるまで待機する。

## 【 0 0 9 3 】

そして、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）では、共有メモリ 5 3 上の同期フラグ 2（Ready Flag）が“0”になったこと、つまり、シミュレーションが完了したことを認識すると（ステップ S 1 6 の Y E S ルート）、共有メモリ 5 3 上の同期フラグ 1（Loop Flag）を“0”に書き換えるとともに（ステップ S 1 7）、シミュレーション結果（センサ信号）である入力データを共有メモリ 5 3 から読み込んでから（ステップ S 1 8）、その入力データを内部メモリ 4 1 5 に書き込む（ステップ S 1 9）。このとき、次の計算時間（処理時間） $T_S$  も書き込まれ、再びステップ S 1 1 に戻って、上述と同様の処理を繰り返し実行する。

#### 【0094】

以上のような処理を繰り返すことにより、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）においてマルチタスクを採用している場合であっても、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）と 3 次元リアルタイムシミュレーション装置 3 0 とを確実に同期させることができる。

なお、このとき、時間  $T_S$  は固定でなく、シミュレーションの状況によって可変にしてもよい。このためには、前述したように、同期とともにシミュレーション時間  $T_S$  を通信する。

#### 【0095】

また、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）において、3 次元リアルタイムシミュレーション装置 3 0 へのデータ（指令信号等）は、3 次元リアルタイムシミュレーション装置 3 0 にシミュレーションを開始させる前に送信され、シミュレーション終了後、3 次元リアルタイムシミュレーション装置 3 0 からのデータ（センサ信号等）は、状態遷移図・表作成編集装置 4 1（マイコンチップ 4 2）側の計算を開始する前に受信され、内部メモリ 4 1 5 に書き出される。

#### 【0096】

さらに、3 次元リアルタイムシミュレーション装置 3 0 とマイコンチップ 4 2 との間におけるインタフェース部 5 2 も、図 5 ～ 図 1 4 により上述したインタフェース部 5 1 と同様に構成されて、このインタフェース部 5 1 と同様に機能し、

マイコンチップ 4 2 においてマルチタスクを採用している場合であっても、マイコンチップ 4 2 と 3 次元リアルタイムシミュレーション装置 3 0 とを確実に同期させることができる。

【 0 0 9 7 】

また、状態遷移図・表作成編集装置 4 1 の具体例としては、例えば、State Flow(MathWorks社製品)マニュアル、StateMate (iLogix社製品) マニュアル、ZIPC (CATS社製品)マニュアルなどに開示された技術が挙げられるほか、統合開発環境装置 4 3 の具体例としては、Softune (富士通社製品) マニュアルなどに開示された技術が挙げられる。これらの状態遷移図・表作成編集装置 4 1 と統合開発環境装置 4 3 とは互いに連携しており、状態遷移図・表作成編集装置 4 1 で作成したタスク制御フローは統合開発環境装置 4 3 にロードされ、この統合開発環境装置 4 3 により、最終的にターゲットマイコン用の実行モジュールがシームレスに作成されるようになっている。

【 0 0 9 8 】

そして、上述のように、メカ設計と組込みソフトウェアの開発とを、3次元リアルタイムシミュレーション装置 3 0 を利用しながらコンカレントに進めた後は、実際のインタフェースの試作品を使って（実機とマイコンチップ 4 2 とを接続して）、メカおよび組込みソフトウェアの結合試験（実機最終確認）を行なう。

またさらに、前述した通り、本実施形態の 3 次元リアルタイムシミュレーション装置 3 0 にはロジックアナライザ（解析部）がそなえられている。このロジックアナライザにおいては、図 1 5 に示すように、アクチュエータ（モータ）に対するアクチュエータ指令信号と 3 次元リアルタイムシミュレーション装置 3 0 で得られたセンサ信号との時間変化が、同期アルゴリズム下で、リアルタイムで解析され、シミュレーション結果として表示される。これにより、組込みソフトウェア開発者がその時間変化を確認することができる。また、ロジックアナライザは、データを保存する機能や印刷する機能も有している。

【 0 0 9 9 】

図 1 6 は、上述したロジックアナライザによる処理を含む、本実施形態でのシミュレーション処理全体の流れを説明するためのフローチャートであり、この図

16に示すように、3次元リアルタイムシミュレーション装置30によるシミュレーション処理中は、インタフェース部51や52による同期処理を行ないながら（ステップS21）、状態遷移図・表作成編集装置41やマイコンチップ42からのアクチュエータ指令信号に基づくモータ処理を行なった後（ステップS22）、そのアクチュエータ指令信号に応じたセンサ処理を行なう（ステップS23）。これらのステップS21～S23の処理中は、I/Oボードやソケット通信による入出力が行なわれ、これらの処理に伴う出力値は、後から読み出すことができるようにバッファリングされる一方、入力値は、同期処理（ステップS21）以外の部分で変化しないことが保証されているため、バッファリングは行なわれていない。

#### 【0100】

そして、ロジックアナライザによる処理が不必要である場合（ステップS24のNOルート）、ステップS21に戻る一方、必要である場合（ステップS24のYESルート）、アクチュエータ指令信号やセンサ信号の記憶・表示が行われる（ステップS25）。

このとき、本実施形態では、ループ毎に、シミュレーションすべき時間 $T_S$ が変化することがあるため、ロジックアナライザの情報は、「時間」，「P I O値」，「D A値」の3つの配列を用いて記憶される。

#### 【0101】

なお、メモリ容量の節約の関係から、「時間」の配列としては、同期処理に際して3次元リアルタイムシミュレーション装置30側から指定される制御ループ回数（図33にて後述するn参照；時間 $T_S$ に対応した回数）を記憶し、「P I O値」の配列としては、各ビット毎に32チャンネル分のデータを記憶し、「D A値」の配列としては、2チャンネル分のデータを16ビットに量子化し32ビットの整数として記憶している。

#### 【0102】

ついで、本実施形態の支援システム10における3次元リアルタイムシミュレーション装置30について、図17～図82を参照しながら、より詳細に説明する。

図 1 7 は、本実施形態の支援システム 1 0 における 3 次元リアルタイムシミュレーション装置 3 0 の動作概念を説明するための概念図であり、この図 1 7 に示すように、3 次元リアルタイムシミュレーション装置（「3 D モデルシミュレータ」と略記する）3 0 内には 3 次元機構モデル（「3 D モデル」と略記する）が構築されている。

#### 【 0 1 0 3 】

3 D モデルは、複数のリンク（製品を構成する個々の部品）の 3 次元形状とそれらのリンクの姿勢との関係を定義した 3 次元機構データによって定義される、製品の 3 次元モデルであり、リンクの姿勢を変更することによってその 3 D モデルが動作することになる。

組込みソフトウェア開発部 4 0（状態遷移図・表作成編集装置 4 1 やマイコンチップ 4 2）から 3 D モデルシミュレータ 3 0 には、モータ等のアクチュエータを動作させるためのアクチュエータ指令信号が送られ、3 D モデルシミュレータから組込みソフトウェア開発部 4 0 には、センサのオン、オフ等をあらわすセンサ信号が送られる。また、組込みソフトウェア開発部 4 0 と 3 D モデルシミュレータ 3 0 との間では同期信号が送受信される。なお、図 1 7 において、同期信号の送受信機能を果たすインタフェース部 5 1，5 2（図 1 参照）の図示は省略されている。

#### 【 0 1 0 4 】

図 1 8 は、センサやモータの定義時の処理を説明するための、3 D モデルシミュレータ 3 0 の模式図、図 1 9 は、本実施形態の 3 D モデルシミュレータ 3 0 によるシミュレーション処理を説明するための模式図である。

3 D モデルは、3 次元 C A D 等のメカ設計部 2 0 により構築され、その構築された 3 D モデルに対し、ここでは、図 1 8 に示すように、ユーザによりキーボード等の入力装置が操作されてモータやセンサが定義される。

#### 【 0 1 0 5 】

そのような定義が行なわれた後の実際のシミュレーションに当たっては、図 1 9 に示すように、組込みソフトウェア開発部 4 0 から送られてきたアクチュエータ指令信号（ここではモータ回転指令信号）に対応し、モータとして定義された



リンク（以下、「モータリンク」と称する）の姿勢を変化させ、さらに機構データに基づき、モータリンクに連結された他のリンクについて、モータリンクの姿勢変化に応じた量だけ動作させ、その結果として、センサとして定義されたリンク（以下、「センサリンク」と称する）の姿勢にも影響が出る。そこで、センサリンクの姿勢に応じセンサ信号を発生させ、そのセンサ信号を組み込みソフトウェア開発部 4 0 に送信する。

#### 【0106】

図 2 0 は、本実施形態の 3 D モデルシミュレータ 3 0 と組み込みソフトウェア開発部 4 0（同期タスク）との動作タイミングを説明するための図であり、この図 2 0 における図中のハッチング部分は、シミュレータ 3 0 や組み込みソフトウェア開発部 4 0 がそれぞれ動作している期間を示している。

この図 2 0 に示すように、シミュレータ 3 0 と組み込みソフトウェア開発部 4 0（同期タスクのオフ動作）とは、動作、停止のサイクルを交互に繰り返している。各サイクルにおいて、シミュレータ 3 0 および組み込みソフトウェア開発部 4 0 は、次のような手順で動作する。ただし、シミュレータ 3 0 と組み込みソフトウェア開発部 4 0（同期タスク）との間の同期の詳細については後述する。

#### 【0107】

（1）シミュレータ 3 0 は、アクチュエータ指令信号を基に、シミュレーション時間  $T_S$  分のシミュレーションを行ない、センサ信号を出力する。

（2）シミュレータ 3 0 は、現在の 3 D モデルの干渉可能性から、次のループでのシミュレーション時間  $T_S$  を決定し、その時間  $T_S$  を組み込みソフトウェア開発部 4 0 に受け渡すとともに、動作開始指令（Ready Flag）を組み込みソフトウェア開発部 4 0 に受け渡す。

#### 【0108】

（3）組み込みソフトウェア開発部 4 0 は、時間  $T_S$  を受け取るとともに動作開始指令（Ready Flag）を受け取り、通り図 1 2 ～図 1 4 を参照しながら前述した通り、同期タスクをオフとして、その時間  $T_S$  分の計算（制御）をマルチタスクで行なった後、シミュレータ 3 0 側に動作終了信号（Loop Flag）を送り、次の動作開始指令（Ready Flag）を受け取るまでの間、同期タスクをオンとして動作

を停止する。

【0109】

(4) シミュレータ30は、組込みソフトウェア開発部40の動作終了信号(Loop Flag)を受け取って時間 $T_S$ 分のシミュレーションを開始する。

シミュレータ30と組込みソフトウェア開発部40とは、以上の手順を繰り返すことにより、交互に同期しながら動作する。

図21は、本実施形態の3Dモデルシミュレータ30によるシミュレーション手順を説明するためのフローチャートである。

【0110】

この図21に示すように、シミュレーションが開始されると、まず初期設定が行なわれて(ステップS101)、3Dモデルの各リンクの姿勢が初期化される。そして、シミュレーションを行なう時間間隔(シミュレーション間隔) $T_S$ が組込みソフトウェア開発部40側に送信され(ステップS102)、組込みソフトウェア開発部40側からシミュレーション間隔 $T_S$ を受信した旨をあらわす信号を受け取ると(ステップS103のYESルート)、組込みソフトウェア開発部40に計算開始命令を送信する(ステップS104)。

【0111】

組込みソフトウェア開発部40によりシミュレーション間隔 $T_S$ 分の計算が行なわれて、その終了が通知されると(ステップS105のYESルート)、ここでシミュレートしている3Dモデルを構成する全てのモータおよび全てのセンサを今回の $T_S$ 分の処理に関し未処理である旨を初期設定する(ステップS106)。

【0112】

次いで、ステップS201では、未処理のモータが存在しているか否か判定され、未処理のモータ1つずつについてステップS202～S207の処理が行なわれる。

すなわち、未処理のモータが存在している場合(ステップS201のYESルート)、組込みソフトウェア開発部40側から送られてきた、今処理しようとしているモータに関するアクチュエータ指令信号に応じてそのモータの回転速度が

決定され（ステップ S 2 0 2）、その回転速度から  $T_S$  間に変位するモータの姿勢が決定される（ステップ S 2 0 3）。

【0 1 1 3】

そして、機構関係定義によりそのモータに関連づけられた他の部品の姿勢が変更され（ステップ S 2 0 4）、部品間の最短距離  $d$  が測定され（ステップ S 2 0 5）、ステップ S 2 0 6、S 2 0 7 では、各モータの回転により測定される各最短距離  $d$  の中の最短距離  $D$  が求められる。

ここでは、今までの最短距離  $D$  よりも新たに測定された最短距離  $d$  が小さいか否かを判断し（ステップ S 2 0 6）、小さい場合（YES ルート）、新たに測定された最短距離  $d$  を最短距離  $D$  に置き換えてから（ステップ S 2 0 7）、ステップ S 2 0 1 に戻る一方、小さくない場合（NO ルート）、そのままステップ S 2 0 1 に戻る。

【0 1 1 4】

上述のようなステップ S 2 0 2 ～ S 2 0 7 の処理は、ステップ S 2 0 1 で NO ルートをとるまで、つまり、未処理のモータが存在しないと判定されるまで、繰り返し実行される。

このようにして、全てのモータについて  $T_S$  間の姿勢変更が終了すると、姿勢変更後の 3 D モデルが画面に表示される（ステップ S 2 0 8）。

【0 1 1 5】

この後、今度は未処理のセンサが存在しているか否か判定され（ステップ S 3 0 1）、未処理センサが存在している場合（YES ルート）、未処理のセンサ 1 つずつについてセンサリンクの姿勢に応じたセンサ信号のオン、オフが検出される（ステップ S 3 0 2）。

全てのセンサについての処理が終了すると（ステップ S 3 0 1 の NO ルート）、組込みソフトウェア開発部 4 0 に向けてそれら全てのセンサに関するセンサ信号が出力され（ステップ S 3 0 3）、さらに、部品間最短距離  $D$  に応じたサンプリング間隔  $T_S$  が決定され（ステップ S 4 0 1）、ステップ S 1 0 2 に戻ってその決定された間隔  $T_S$  が組込みソフトウェア開発部 4 0 に送信される。

【0 1 1 6】

シミュレータ 3 0 では、以上の処理が繰り返し実行され、組込みソフトウェア開発部 4 0 との同期をとりつつ、3 D モデルの動作シミュレーションが行なわれる。

図 2 2 は、図 2 1 に示すシミュレータの処理フローの一部分（図 2 1 に一点鎖線で囲った部分）に代えて採用することのできる部分フローを示すフローチャートである。

#### 【 0 1 1 7 】

この図 2 2 には、ステップ S 2 0 2 とステップ S 2 0 3 との間にステップ S 2 1 0 が挿入されている。このステップ S 2 1 0 では、今処理を行なっているモータの回転速度がゼロ ( $\omega = 0$ ) か否かが判定され、今回の間隔  $T_S$  の間にそのモータは回転しない ( $\omega = 0$ ) のときは (YES ルート)、そのモータの動作に関しては部品間の距離  $d$  の測定は行なわずにステップ S 2 1 1 に進んで  $d = \infty$  としてからステップ S 2 0 6 へ移行し、このステップ S 2 0 6 の判定でその距離  $d = \infty$  が距離  $D$  を求める演算から除外されるようにしている。ステップ S 2 1 0 で今処理を行なっているモータの回転速度がゼロではないと判定された場合 (NO ルート)、ステップ S 2 0 3 へ移行し、前述した通りの処理を行なう。

#### 【 0 1 1 8 】

部品間の距離  $d$  を求めるにはかなりの計算量を要するので、上述のごとく、モータが回転しないときはそのモータに関する距離  $d$  の測定を行なわないようにすることにより、シミュレータ 3 0 での計算量を減らし、シミュレーションのスピードアップをはかることができる。

図 2 3 は、モータとして定義されるリンク（モータリンク）の選び方を説明すべくモータリンクの一例を模式的に示す斜視図である。

#### 【 0 1 1 9 】

この図 2 3 において、本来のモータは図中のリンク A であるが、リンク A 自体は土台に固定されているため、シミュレーション上は、その本来のモータによって一番最初に駆動されて 3 D モデルで姿勢が変更されるリンク B を、モータリンクとして定義する。モータリンクをこのように定義することにより、シミュレーションを容易化している。

## 【0 1 2 0】

図 2 4 (A), (B) および図 2 5 (A), (B) は、センサとして定義されるリンク（センサリンク）の選び方を説明すべく、それぞれ、センサリンクの一例および他例を模式的に示す斜視図である。

これらの図 2 4 (A), (B) および図 2 5 (A), (B) において、本来のセンサはリンク A であるが、ここでは、その本来のセンサによって検出されるリンク B をセンサリンクとして定義する。そして、リンク B の姿勢（位置）に応じて、図 2 4 (A) および図 2 5 (A) に示す状態を、センサがオフの状態として定義し、図 2 4 (B) および図 2 5 (B) に示す状態を、センサがオンの状態として定義する。このように、センサの場合も、モータの場合と同様、本来のセンサではなくそのセンサにより検出されるリンクをセンサとして定義し、そのセンサリンクの姿勢に応じてオン、オフを定義することにより、シミュレーションを容易化している。

## 【0 1 2 1】

図 2 6 は、モータの種類を説明するための図である。

この図 2 6 に示すように、本実施形態では、モータを定義するにあたり「無段変速」と「有段変速」との二種類の定義方法が用意されている。

「無段変速」は、速度指令値と目標速度との間の比例定数を定義する定義方法である。この「無段変速」は、さらに、「正負に関係なく比例」させる方法と、二進数の「最上位ビットを符号」とみなす方法との 2 種類が用意されている。前者の方法では、例えば比例定数が 1 0 0 のとき、二進数 1 0 1（1 0 進数で -3）が -3 0 0 r p m（マイナス符号は逆転方向への回転をあらわす）に変換される。また、後者の方法では、例えば二進数 1 0 1 について、最上位ビット '1' とその他のビット '0 1' とに分け、最上位ビット '1' はマイナス符号（'0' はプラス符号）、残りの '0 1' を数値（1 0 進数で 1）とし、それらを合わせた -1 に比例定数 1 0 0 を掛けて -1 0 0 r p m に変換する。

## 【0 1 2 2】

「有段変速」は、速度指令値と目標速度との対応表が定義される。例えば図 2 6 中の対応表では、二進数 1 0 1 に対し -2 0 0 r p m が定義されており、速度

指令値として二進数 1 0 1 が入力されると、対応表を参照し目標速度として - 2 0 0 r p m が出力される。なお、対応表としては、図 7 5 を参照しながら後述するようなものもある。

#### 【 0 1 2 3 】

本実施形態では、上述のような複数種類の定義方法を用意しておくことにより、モータの定義の自由度を向上させている。

図 2 7 は、モータを一次遅れ系とみなしたときの、モータの回転速度  $\omega$  の初期変化を示す図である。

ここでは、モータの回転速度変化を図 2 7 に示すような一次遅れ系としてとらえて、モータの負荷による遅れやその影響を取り入れたシミュレーションを行なう。モータを一次遅れ系とみなすと、ユーザは、整定時間  $4 T$  を入力するだけでそのモータの特性を指定することができ、入力作業の繁雑さが低減される。

#### 【 0 1 2 4 】

以下に、モータを一次遅れ系とみなしたときの、シミュレーション間隔  $T_S$  の間のモータの回転量  $\Delta \theta$  の算出式を導出する。なお、以下、シミュレーション間隔  $T_S$  を  $\Delta t$  として表記する。

時刻  $t$  におけるモータ速度  $\omega$  は、下式 (1) により表される。

#### 【 0 1 2 5 】

##### 【数 1】

$$\omega = \omega_o + (\omega_i - \omega_o) \left( 1 - e^{-\frac{t}{T}} \right) \quad (1)$$

#### 【 0 1 2 6 】

ここで、 $\omega_i$  は目標速度、 $\omega_o$  は目標速度指令前のモータ速度、 $T$  はモータの性質を表す整定時間  $4 T$  に比例した定数である。

上式 (1) で表される時刻  $t$  におけるモータ速度  $\omega$  を  $\omega_{k-1}$  とすると、時刻  $t$  よりも  $\Delta t$  後の時刻  $t + \Delta t$  のモータ速度  $\omega_k$  は、下式 (2) となる。

#### 【 0 1 2 7 】

【数 2】

$$\omega_k = \omega_o + (\omega_i - \omega_o) \left( 1 - e^{-\frac{t+\Delta t}{T}} \right) \quad (2)$$

【0128】

これらの (1), (2) 式から下式 (3) が得られる。

【0129】

【数 3】

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_{k-1}) \left( 1 - e^{-\frac{\Delta t}{T}} \right) \quad (3)$$

【0130】

$\Delta t$  の間にモータが回転する量  $\Delta \theta$  は、上式 (3) を  $\Delta t$  について積分することにより、下式 (4) として与えられる。

【0131】

【数 4】

$$\begin{aligned} \Delta \theta &= \int_0^{\Delta t} \left[ \omega_{k-1} + (\omega_i - \omega_{k-1}) \left( 1 - e^{-\frac{t}{T}} \right) \right] dt \\ &= \omega_{k-1} \Delta t + (\omega_i - \omega_{k-1}) \left\{ \Delta t - T \left( 1 - e^{-\frac{\Delta t}{T}} \right) \right\} \end{aligned} \quad (4)$$

【0132】

モータを一次遅れ系とみなしたとき、上式 (4) に基づいて、目標速度  $\omega_i$ 、シミュレーション時間  $\Delta t$ 、そのシミュレーション時間  $\Delta t$  だけ前の時刻におけるモータ速度  $\omega_{k-1}$  および整定時間  $4T$  によって、そのシミュレーション時間  $\Delta t$  の間のモータの回転量  $\Delta \theta$  が求められる。

ただし、モータは、常に上記 (4) 式のように動作する必要はなく、例えばモータに指令される目標速度  $\omega_i$  が変化してから整定時間  $4T$  が経過した時点で速度を  $\omega_i$  に固定してもよい。

## 【 0 1 3 3 】

図 2 8 は、本実施形態のシミュレータ 3 0 における、一次遅れを考慮したモータ駆動の処理を説明するためのフローチャートである。

この図 2 8 に示すように、部品間距離に基づいてシミュレーション間隔  $\Delta t$  ( $=T_s$ ) が決定され (ステップ S 5 0 1)、組込みソフトウェア開発部 4 0 からアクチュエータ指令信号が入力されると (ステップ S 5 0 2)、図 2 6 に示すモータの定義に従って比例定数あるいは対応表が参照されて目標速度  $\omega_i$  が決定される (ステップ S 5 0 3)。

## 【 0 1 3 4 】

ステップ S 5 0 4 では、目標速度  $\omega_i$  が変更されたか否か ( $\omega_i = \omega_{iold}$  であるか否か) が判定され、今回の目標速度  $\omega_i$  が変更されていた場合 (N O ルート;  $\omega_i \neq \omega_{iold}$ ) は、ステップ S 5 0 5 に進み、目標速度  $\omega_i$  が変更された時点からの経過時間を示す  $t_{total}$  が 0 に初期化され、次のステップ S 5 0 4 での判定のために  $\omega_{iold}$  に  $\omega_i$  が格納されてから (ステップ S 5 0 6)、ステップ S 5 0 7 へ移行する。

## 【 0 1 3 5 】

ステップ S 5 0 4 で  $\omega_i = \omega_{iold}$  であると判定された場合 (Y E S ルート) は、ステップ S 5 0 5、S 5 0 6 を経由することなく、ステップ S 5 0 7 に進み、目標速度  $\omega_i$  が変更された時点からの経過時間  $t_{total}$  が整定時間  $4 T$  に達したか否かが判定される。 $t_{total}$  が未だ  $4 T$  に達していない場合 (N O ルート) は、ステップ S 5 0 8 に進み、上式 (3) に従ってモータの回転速度  $\omega$  が求められ、さらにステップ S 5 0 9 において、上式 (4) に従ってモータの回転量  $\Delta \theta$  が求められる。

## 【 0 1 3 6 】

一方、ステップ S 5 0 7 において  $t_{total} \geq 4 T$  であると判定された場合 (Y E S ルート) は、ステップ S 5 1 0 に進み、モータの回転速度  $\omega$  が目標速度  $\omega_i$  と等しい速度に設定され、さらに、ステップ S 5 1 1 において、モータの回転量  $\Delta \theta$  が、目標速度  $\omega_i$  で時間  $\Delta t$  だけ回転したときの回転量に設定される。

ステップ S 5 0 8、S 5 0 9 を経由した場合であっても、あるいはステップ S



5 1 0, S 5 1 1 を経由した場合であっても、ステップ S 5 1 2 において  $t_{total}$  が  $\Delta t$  だけ増分され、ステップ S 5 1 3 においてモータのトータルの回転量  $\theta$  が  $\Delta \theta$  だけ増分される。

#### 【 0 1 3 7 】

この図 2 8 に示す処理が組込みソフトウェア開発部 4 0 と同期をとりながら繰り返し実行されることにより、図 2 7 に示す一次遅れ系とみなしたときの速度変化に従ってモータが回転する。

図 2 9, 図 3 0 および図 3 1 (A) ~ 図 3 1 (C) は、干渉チェックを用いたセンサの定義方法を説明するための図である。

#### 【 0 1 3 8 】

図 2 9 では、接触センサ S 1 の姿勢が複数のモータ M の姿勢の組合せにより決定される例が示されている。この場合、図 2 4 (A), (B) および図 2 5 (A), (B) を参照して説明した方法を採用すると、複数のモータ M それぞれの姿勢の組合せでセンサ S 1 のオン、オフを定義することになり極めて複雑な定義が必要となる。そこで、ここでは、センサ S 1 と土台との干渉をチェックし、それらの間の干渉の有無によってセンサ S 1 をオンあるいはオフとする。こうすることにより、ユーザに複雑な定義を要求することなく、センサ S 1 のオン、オフを決定することができる。

#### 【 0 1 3 9 】

図 3 0 では、スリット S L の移動により光量センサ（光電センサ） S 2 のオン、オフが多数回繰り返される例が示されており、この場合、図 3 1 (A) ~ 図 3 1 (C) に示すように光束を 1 つの仮想リンクと見なし、その仮想リンクとスリット S L との干渉の有無によってセンサ S 2 のオン、オフを判定する。

この場合、仮想リンクとの干渉をチェックする相手のリンクは、図 3 0 に示すように相手のリンクが予め分かっている場合、そのリンクに限定してもよい。また、相手のリンクが不明である場合であっても、本来のセンサ自体をあらわすリンクは、その仮想リンクと常に干渉（接触）しているため、干渉をチェックすべき相手のリンクからは除外される。

#### 【 0 1 4 0 】

この仮想リンクによるセンサ出力決定方法は、ここに示した光電センサなどに限らず、例えばマイクロスイッチの接触部分を仮想リンクとして定義するなど、接触式の位置検出センサにも適用することができる。

図 2 4 (A), (B) および図 2 5 (A), (B) を参照して説明したセンサの定義方法は、ポテンショメータ、エンコーダ等の角度を検出するセンサなど、一自由度の変位を検出するセンサを表現するのに有効な方法である。しかし、ここに示したような、複数のモータの姿勢の組合せにより出力が決定されるように配置されたセンサ (図 2 9) など多自由度の組合せの結果として姿勢が決定されるセンサや、スリットカウンタ (図 3 0) などオン、オフの変化が多いセンサには、その定義方法を採用するとユーザに複雑な定義を要求することになる。そのため、ここに説明した、リンクどうしの干渉の有無によりセンサのオン、オフを判定する方法を採用することが好ましい。

#### 【 0 1 4 1 】

図 3 2 は、シミュレータ 3 0 と組込みソフトウェア開発部 4 0 との間で同期をとるために送受信される信号を説明するためのタイミングチャート、図 3 3 は、シミュレータ 3 0 における 3 D モデルの動作とソフトウェア開発部 4 0 の動作との同期をとるための同期化手法を説明するためのフローチャート、図 3 4 は、その同期を実現するための、シミュレータ 3 0 側の処理を説明するためのフローチャートである。ここで、図 3 3 で説明する同期化手法を実現するためのプログラムは、インタフェース部 5 1, 5 2 を実現するための第 2 インタフェースプログラム 3 1 5 に組み込まれる。

#### 【 0 1 4 2 】

図 3 2 に示す、ループフラグ (Loop Flag), レディフラグ (Ready Flag) およびタイムスケールは、それぞれ以下の意味をもつ信号である。

##### (1) Loop Flag

組込みソフトウェア開発部 4 0 からシミュレータ 3 0 へ受け渡す信号であって、組込みソフトウェア開発部 4 0 が現在動作中であるか否かを表わしている。

#### 【 0 1 4 3 】

##### (2) Ready Flag

シミュレータ 3 0 から組込みソフトウェア開発部 4 0 へ受け渡す信号であって、シミュレータ 3 0 による 1 つのシミュレーション間隔  $\Delta t$  ( $=T_S$ ) 分のシミュレーションが終了したことを表している。

### (3) タイムスケール

シミュレータ 3 0 から組込みソフトウェア開発部 4 0 へ受け渡す信号であって、シミュレータ 3 0 が部品間最短距離に基づいて決定したシミュレーション間隔  $\Delta t$  ( $=T_S$ ) を表している。

#### 【0 1 4 4】

本実施形態では、シミュレーション間隔  $\Delta t$  ( $=T_S$ ) を指定するタイムスケールとしては、5 m s e c, 2 s e c などの具体的な時間を用いず、そのシミュレーション間隔  $\Delta t$  ( $=T_S$ ) の間に、図 3 3 に示す処理ルーチンを通過する回数  $n$  を用いている。

図 3 2 に示すように、ステップ S 6 0 1 で  $n = 0$  か否かを判定し、 $n = 0$  でない場合 (N O ルート) には、ステップ S 6 0 2 に進んで  $n$  を 1 だけデクリメントし、本来の処理 (遷移図・表作成作成プログラム 3 2 0 や制御プログラム) の実行に移っている。そして、ステップ S 6 0 1 において  $n = 0$  と判定された場合 (Y E S ルート) のみ、Loop Flag を 'L o w' に設定し同期タスクを起動させてから (ステップ S 6 0 3)、ステップ S 6 0 4 ~ S 6 0 7 の処理が実行される。

#### 【0 1 4 5】

ここでは、図 3 2 ~ 図 3 4 を説明するに当たり、シミュレータ 3 0 では、現在、シミュレーションが行なわれており、従って、Ready Flag は 'L o w' レベルであり、組込みソフトウェア開発部 4 0 側では図 3 3 のステップ S 6 0 4 にとどまって Ready Flag が 'H i g h' に変化するを待っている段階にあるとする。このとき、シミュレータ 3 0 における同期処理をあらわす図 3 4 のルーチンでは、ステップ S 7 0 3 のシミュレーションを行なっている途中にある。

#### 【0 1 4 6】

なお、シミュレータ 3 0 側では、初期設定を行なってから (ステップ S 7 0 1)、ステップ S 7 0 2 に進んで、Loop Flag が 'L o w' に変化するのを待ち、前述した図 3 2 のステップ S 6 0 3 の処理により Loop Flag が 'L o w' に変化

すると、ステップ S 7 0 3 に進み、シミュレーション間隔  $\Delta t$  分のシミュレーションが行なわれる。

【0 1 4 7】

ここで、シミュレーションが終了すると、シミュレータは図 3 4 のステップ S 7 0 4 に移り、次のシミュレーション間隔（タイムスケール） $\Delta t$  が、上述した回数  $n$  により決定され、ステップ S 7 0 5 においてそのタイムスケール  $\Delta t$  ( $n$ ) が組込みソフトウェア開発部 4 0 に送信され、さらにステップ S 7 0 6 に進んで Ready Flag が 'High' レベルに変更される。

【0 1 4 8】

これに応じて、組込みソフトウェア開発部 4 0 側では、図 3 3 に示すステップ S 6 0 4 を抜け出し（YES ルート）、ステップ S 6 0 5 に進みタイムスケール  $n$  が読み込まれ、さらにステップ S 6 0 6 に進んで Loop Flag が 'High' に変更され同期タスクが停止される。その後、ステップ S 6 0 7 において、シミュレータ 3 0 側から送信されてきている Ready Flag が 'Low' に変化するのを待つて本来の処理に進む。

【0 1 4 9】

組込みソフトウェア開発部 4 0 において遷移図・表作成作成プログラム 3 2 0 や制御プログラムは一定周期で繰り返し実行されており、前述したように、ステップ S 6 0 1 では  $n = 0$  か否かが判定され、 $n \neq 0$  の場合（NO ルート）、ステップ S 6 0 2 に進んで  $n$  が 1 だけデクリメントされる。これを繰り返し、ステップ S 6 0 1 で  $n = 0$  と判定されると（YES ルート）、ステップ S 6 0 3 に進み Loop Flag が 'Low' に変更されて同期タスクが起動され、その後、ステップ S 6 0 4 において、シミュレータ 3 0 側から送られてくる Ready Flag が 'High' に変化するのを待つ状態となる。

【0 1 5 0】

一方、シミュレータ 3 0 側では、図 3 4 のステップ S 7 0 6 において Ready Flag が 'High' に変更された後、ステップ S 7 0 7 において、組込みソフトウェア開発部 4 0 側から送られてくる Loop Flag が 'High' に変化するのを待ち、前述した図 3 2 のステップ S 6 0 6 の処理により Loop Flag が 'High' に

変化したことを受けて、ステップ S 7 0 8 に進み、Ready Flag を 'L o w' に変化させ、ステップ S 7 0 2 に進んで、Loop Flag が 'L o w' に変化するのを待つ。前述したように、Loop Flag が 'L o w' に変化すると、ステップ S 7 0 3 に進み、シミュレーション間隔  $\Delta t$  分のシミュレーションが行なわれる。

#### 【0 1 5 1】

以上の処理が繰り返されることにより、シミュレータ 3 0 と組込みソフトウェア開発部 4 0（遷移図・表作成プログラム 3 2 0 や制御プログラム）は相互に同期をとられながら、それぞれの処理がマルチタスクにより実行される。

次に、リンク（部品）どうしの干渉可能性の評価方法について説明する。

図 3 5（A）および図 3 5（B）は、干渉可能性の評価方法の説明図である。ここでは、リンク A が速度  $v$  でリンク B に向かって動いている例が示されている。

#### 【0 1 5 2】

図 3 5（A）は、干渉する可能性のある 2 つのリンク（リンク A とリンク B）が大きく離れている状態を示している。この場合、干渉可能性が小さいものと判定されて、シミュレーション間隔  $\Delta t$  として比較的大きな値を設定し、リンク A が一回のシミュレーションで進む距離  $v \Delta t$  を大きくすることにより、シミュレーションを高速に行なうことができる。

#### 【0 1 5 3】

また、図 3 5（B）は、リンク A とリンク B とが近づき、それらの間の距離が小さい状態を示している。この場合、干渉可能性が大きいものと判定されて、シミュレーション間隔  $\Delta t$  として比較的小さな値を設定し、リンク A が一回のシミュレーションで進む距離  $v \Delta t$  を小さくすることにより、高精度なシミュレーションを行なうことができる。

#### 【0 1 5 4】

このように、本実施形態では、部品間の距離に応じてシミュレーション間隔  $\Delta t$  が変更され、高精度なシミュレーションと高速なシミュレーションとの両立、調和が図られている。

図 3 6（A）～図 3 6（C）は、部品間の距離  $d$  とシミュレーション間隔  $\Delta t$

との関係の各種の例を示す図である。

【0 1 5 5】

これらの図 3 6 (A) ～図 3 6 (C) に示すいずれの例によっても、距離  $d$  が小さくてもシミュレーションがあまりにも低速化されないように、且つ、距離  $d$  が大きくてもシミュレーションがあまりにも高速化されないように、部品間の距離  $d$  に対するシミュレーション間隔  $\Delta t$  が決定される。

次に、干渉可能性の評価のために部品をグループ分けする方法について説明する。

【0 1 5 6】

図 3 7 (A) および図 3 7 (B) は、部品のグループ分けを説明するための図である。

これらの図 3 7 (A) および図 3 7 (B) においては、リンク  $l_0$  ～リンク  $l_5$  の 6 つのリンクが示されており、モータリンクとして定義されたリンク  $l_3$  が動く場合を考える。

【0 1 5 7】

この場合、モータリンク  $l_3$  の動きによって姿勢に影響を受けるリンクはリンク  $l_3$  ～ $l_5$  であり、これらのリンク  $l_3$  ～ $l_5$  は、モータリンク  $l_3$  の動きに対して 1 つの剛体と考えることができる。

また、モータリンク  $l_3$  の動きによっては姿勢に影響を受けないリンク  $l_0$  ～ $l_2$  も同様に、これらを合わせたものを 1 つの剛体と考えることができる。

【0 1 5 8】

そこで、モータリンク  $l_3$  を動作させる場合には、図 3 7 (B) に示すように、そのモータリンク  $l_3$  を動かしたことにより姿勢に影響を受けるリンクのグループと影響を受けないリンクのグループとの 2 つのグループに分け、それら 2 つのグループそれぞれを剛体とみなし、それら 2 つの剛体の間の最短距離を求めることにより、その求めた距離を次のシミュレーション間隔  $\Delta t$  を決定するための最短距離の候補の 1 つとすることができる。

【0 1 5 9】

ただし、図 3 7 (A) および図 3 7 (B) におけるリンク  $l_3$  とリンク  $l_2$  (例

えばモータの軸と歯車等)は、常に接触しているにも拘らず異なるグループに分けられてしまい、このままでは上記2つの剛体は常に干渉していることになってしまう。

そこで、本実施形態では、上記のようにして2つのグループに分けた後、一度、部品間距離を求め、既に接触・干渉しているリンクのうち、今注目しているモータリンクを動かしたときに動かない方のグループに属するリンク〔図37(A)および図37(B)に示す例の場合リンク $l_2$ 〕をそのグループから除外した上で2つのグループそれぞれを剛体とみなし、それら2つの剛体の間の距離が求められる。

#### 【0160】

図38(A)～図38(D)は、1つの3Dモデル内に複数のモータリンクが定義されている場合のグループ分けを説明するための図である。

図38(A)は、3Dモデルをあらわしており、ここには、モータM1～モータM3の3つのモータリンクが定義されている。

このとき、図37(A)および図37(B)を参照して説明したグループ分けを、モータM1のみを動かした時〔図38(B)参照〕と、モータM2のみを動かした時〔図38(C)参照〕と、モータM3のみを動かしたとき〔図38(D)参照〕とのそれぞれについて行なう。

#### 【0161】

そして、シミュレーション実行時には、対象としているモータリンクを動かしたときは、そのモータリンクに関しグループ分けしたグループ間の最短距離が求められ、それら複数のモータリンクの動きにより求められた複数の‘最短距離’の中からさらに一番最短の距離が求められる(図21のステップS201～S207参照)。

#### 【0162】

図39は、グループ分けした部品の再グループ化を説明するための図である。

図38(A)～図38(D)を参照して説明した手法によると、各モータリンクに関し、その3Dモデルを構成するほぼ全ての部品がどちらかのグループ(剛体)に属する結果となり、この場合、各剛体を定義するためのデータ量が膨大な

ものとなってしまう、極めて大きなメモリ容量を必要とする可能性がある。

【0163】

そこで、図39に示すように、各モータリンクの動きによって動かない方のグループ〔図39に示す例ではグループ(D)，(E)，(F)〕に属する部品群のうち、共通部分を抜き出して1つの剛体とするとともに、各グループ(D)，(E)，(F)からその共通部分を除いた部分をそれぞれ1つの剛体とする。そして、距離の測定の際には、共通部分からなる剛体とその共通部分を除いた部分の剛体とを合体させて、相手のグループ〔グループ(D)，(E)，(F)〕に対応してそれぞれグループ(A)，(B)，(C)〕との間の距離を求めるようにしてもよい。これにより、共通部分を各グループに配置しておく場合〔図38(A)～図38(D)の場合〕と比べ、これらのグループを定義するデータを格納しておくためのメモリの容量を減らすことができる。

【0164】

図40は、複数のリンクを、モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを説明するためのフローチャート(ステップS801～S811)、図41は、図40に示すのサブルーチンである検索ルーチンを説明するためのフローチャート(ステップS901～S911)である。

【0165】

ここでは、モータリンクの動きによって姿勢に影響を受けるリンクがリストAにリストアップされ、それ以外のリンクがリストBにリストアップされる。なお、リストAおよびリストBは、ここで説明するフローの実行に先立って白紙状態にクリアされているものとする。

図40に示すように、まず、リンクを指し示すポインタPによりモータリンク(現リンク)が指し示され(ステップS801)、そのポインタPで指し示されるリンクがリストAに追加される(ステップS802)。

【0166】

そして、ポインタPで指し示されているリンクにチャイルド(child)のリンクが存在しているか否かが判定される(ステップS803)。



ここで、チャイルドとは、例えば図 3 7 (A) および図 3 7 (B) に示すリンク機構においてポインタがモータリンク 1<sub>3</sub> を指し示していたとき、そのモータリンク 1<sub>3</sub> が動くことによって一緒に動くリンクの 1 つ (ここではリンク 1<sub>4</sub>) をいう。この場合、リンク 1<sub>4</sub> が仮にモータリンクであったとしても今着目しているモータリンクはリンク 1<sub>3</sub> であり、リンク 1<sub>4</sub> はリンク 1<sub>3</sub> のチャイルドとなる。なお、ポインタ P が指し示すリンクと同格の複数のチャイルドが考えられるときであっても、チャイルドはそのうちの 1 つに限定され、チャイルドと考えられる他のリンクは、その 1 つの限定されたチャイルドのブラザ (brother) として定義される。

## 【 0 1 6 7 】

ポインタ P が指し示すリンクにチャイルドがあったときは (ステップ S 8 0 3 の YES ルート)、ステップ S 8 0 4 に進み、そのチャイルドのリンクを指し示すようにポインタ P を移し、図 4 1 に示す検索ルーチンが呼び出される (ステップ S 8 0 5)。その検索ルーチンについては後述する。ステップ S 8 0 5 での検索ルーチンの実行を終了すると、ステップ S 8 0 6 に進む。なお、ステップ S 8 0 3 において、ポインタ P が指し示すリンクにチャイルドが存在しないと判定された場合 (NO ルート) は、直接、ステップ S 8 0 6 に進む。

## 【 0 1 6 8 】

ステップ S 8 0 6 では、ポインタ P で指し示されたリンクが、ギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定される。

ポインタ P が受動部品を駆動する駆動リンクであったときは (YES ルート)、ステップ S 8 0 7 に進み、ポインタ P が指し示すリンクによって駆動される受動部品であるリンクを指し示すようにポインタ P が変更され、ステップ S 8 0 8 において図 4 1 の検索ルーチンが呼び出される。ステップ S 8 0 8 での検索ルーチンの実行を終了すると、ステップ S 8 0 9 に進む。なお、ステップ S 8 0 6 において、ポインタ P が指し示すリンクが受動部品を駆動するリンクではないと判定された場合 (NO ルート)、直接、ステップ S 8 0 9 に進む。

## 【 0 1 6 9 】

このステップ S 8 0 9 まで進んだ段階では、リスト A が完成しており、このス

テップ S 8 0 9 では、全リンクからリスト A にリストアップされたリンクを除いたリンクがリスト B に格納される。

次いで、リスト A にリストアップされた全てのリンクとリスト B にリストアップされた全てのリンクとの間で干渉チェックが行なわれ（ステップ S 8 1 0）、相互に干渉し合っていたリンクの組のうち、リスト B にリストアップされた方のリンクがリスト B から削除される（ステップ S 8 1 1）。

【 0 1 7 0 】

図 4 1 に示す検索ルーチンでは、まず、ステップ S 9 0 1 において、ポインタ P が指し示すリンクがリスト A に既にリストアップされているか否かが判定され、既にリストアップされていた場合（YES ルート）は、そのままこの検索ルーチンを抜けてこの検索ルーチンが呼ばれた元のルーチンに戻る。一方、ステップ S 9 0 1 において、ポインタ P が指し示すリンクがリスト A にリストアップされていないと判定された場合（NO ルート）、ステップ S 9 0 2 に進み、ポインタ P が指し示すリンクがリスト A に追加される。

【 0 1 7 1 】

そして、ステップ S 9 0 3 でポインタ P が指し示すリンクにチャイルドが存在するか否かが判定され、チャイルドが存在していた場合（YES ルート）、ステップ S 9 0 4 に進んでポインタ P がその c h i l d のリンクを指し示すように変更され、ステップ S 9 0 5 において再度この図 4 1 に示す検索ルーチンが呼び出される。このように、この検索ルーチンは何重にも重なって呼び出されることがあり、この検索ルーチンの実行を終了すると、その検索ルーチンを呼び出したステップ（例えばステップ S 9 0 5 で呼ばれたときはそのステップ S 9 0 5）に戻る。ステップ S 9 0 5 での検索ルーチンの実行を終了すると、ステップ S 9 0 6 に進む。なお、ステップ S 9 0 3 において、ポインタ P が指し示すリンクにチャイルドが存在しないと判定された場合（NO ルート）、直接、ステップ S 9 0 6 に進む。

【 0 1 7 2 】

ステップ S 9 0 6 では、ポインタ P が指し示すリンクにブラザが存在するか否かが判定され、ブラザが存在していた場合（YES ルート）、ステップ S 9 0 7

に進んでポインタ P がそのブラザのリンクを指し示すように変更され、ステップ S 9 0 8 において再度この検索ルーチンが呼び出される。ステップ S 9 0 8 での検索ルーチンの実行を終了すると、ステップ S 9 0 9 に進む。なお、ステップ S 9 0 6 において、ポインタ P が指し示すリンクにブラザが存在しないと判定された場合（N O ルート）、直接、ステップ S 9 0 9 に進む。

#### 【 0 1 7 3 】

ステップ S 9 0 9 では、ポインタ P で指し示されるリンクが、ギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定され、駆動リンクであった場合（Y E S ルート）、ステップ S 9 1 0 に進んでポインタ P がその駆動リンクを指し示すように変更され、ステップ S 9 1 1 においてこの検索ルーチンが再度呼び出される。ステップ S 9 1 1 での検索ルーチンの実行を終了すると、検索ルーチンを呼び出したステップに戻る。なお、ステップ S 9 0 9 において、ポインタ P が指し示すリンクが受動部品を駆動するリンクではないと判定された場合（N O ルート）、直接、検索ルーチンを呼び出したステップに戻る。

#### 【 0 1 7 4 】

図 4 2 は、図 4 0 および図 4 1 に示すルーチンの動作説明用のリンク機構モデルを模式的に示す斜視図、図 4 3 は、図 4 2 に示すリンク機構モデルのデータ構造を説明するための図である。

図 4 2 に示すように、リンク A はモータ、リンク B およびリンク C は、それぞれ、リンク A (モータ) の回転に伴って一緒に回転するシャフトおよびギア、リンク D はリンク C と噛合しているギアである。また、リンク E はリンク D (ギア) の回転に伴って一緒に回転するシャフト、リンク F はリンク D (ギア) と噛合しているギアである。

#### 【 0 1 7 5 】

データ構造上は、図 4 3 に示すように、リンク A とリンク D とリンク F とが同格に並び、リンク B はリンク A のチャイルド、リンク C はリンク B のブラザ、リンク E はリンク D のチャイルドの関係にある。

図 4 2 および図 4 3 に示すリンク機構モデルについて、図 4 0 および図 4 1 に示すルーチンの実行によりリスト A を作成することを考える。ここでは、リンク

A（モータ）から検索を始める。すなわち、まず、図40のステップS801においてリンクAが指し示されるようにポインタPが指定される。以下の〔〕内の大文字アルファベットは、リストAにリストアップされるリンクをあらわしている。

【0176】

- （1）ポインタPがリンクAを指し示す状態でスタートする。〔A〕
- （2）ポインタPがリンクAからそのチャイルドであるリンクBに変更され、検索ルーチンが実行される。〔A，B〕
- （3）ポインタPがリンクBからブラザであるリンクCに変更され、検索ルーチンが実行される。〔A，B，C〕
- （4）ポインタPがリンクCの受動部品であるリンクDに変更され、検索ルーチンが実行される。〔A，B，C，D〕
- （5）ポインタPがリンクDからそのチャイルドであるリンクEに変更され、検索ルーチンが実行される。〔A，B，C，D，E〕
- （6）リタンでポインタPがリンクDに戻る。

【0177】

- （7）ポインタPがリンクDからその受動部品であるリンクFに変更され、検索ルーチンが実行される〔A，B，C，D，E，F〕
- （8）リタンでポインタPがリンクDに戻る。
- （9）リタンでポインタPがリンクCに戻る。
- （10）リタンでポインタPがリンクBに戻る。

【0178】

- （11）リタンでPがリンクAに戻る。
- （12）終了

以上の手順を踏んで、図42に示す全てのリンク（リンクA～リンクF）がリストAにリストアップされる。

次に、1つのモータリンクの動きによって姿勢の変化を受けるグループ内での干渉可能性の評価方法について説明する。

【0179】

モータリンクの動きによって干渉する可能性があるのは、図 3 7 (A) および図 3 7 (B) を参照して説明したような、そのモータリンクの動きによって姿勢に影響を受けるリンクのグループと姿勢に影響を受けないリンクのグループとの間だけでなく、図 4 2 に示すように姿勢に影響を受けるグループの内部でも干渉が生じる可能性がある。

#### 【0 1 8 0】

そこで、そのグループの内部で干渉する可能性のあるリンクの組を発見するために、ここでは、シミュレーションに先立って、モータリンクを少しずつ動かしながら、そのモータリンクを動かすことによって姿勢に影響を受けるグループ内の全てのリンクどうしの間の干渉チェックを行なう。このような総当たりの干渉チェックによって干渉したリンクの組は実際のシミュレーションにあたっても相互間の距離を求める対象に加え、シミュレーション間隔  $\Delta t$  ( $=T_s$ ) を決定するための最短距離の候補の 1 つとする。これにより、正確なシミュレーションを行なうことができる。

#### 【0 1 8 1】

ただし、モータリンクを動かしたときに姿勢に影響を受けるグループ内において、例えば図 4 2 に示すモータや歯車のように、常に接触しているリンクの組も存在する。そこで、モータリンクを少しずつ動かして干渉チェックを行なった際に、そのモータリンクの全ての姿勢で常に干渉・接触していたリンクの組、例えば歯車等で関係づけられたリンクの組はリンク間の距離を求める対象から排除される。

#### 【0 1 8 2】

図 4 4 (A) ～図 4 4 (D) は、モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法を具体的に説明するための図である。

図 4 4 (A) は、ある 1 つのモータの動きによって姿勢に変化を受けるグループを構成する全部品を示しており、ここでは一例として、図 4 4 (B) に示すように、そのモータを  $-180^\circ$  から  $180^\circ$  まで  $36^\circ$  おきに回転させ、各回転ごとに干渉チェックを行なうこととする。この場合、モータの回転およびリンク

内の干渉チェックは 1 1 回行なわれる。

【0 1 8 3】

その結果、例えば図 4 4 (C) に示すような、干渉したリンクの組とその干渉回数とが得られたものとする。1 1 回の干渉チェックを行なったところ、リンク B とリンク E との干渉、および、リンク D とリンク E との干渉は 1 1 回発生（常に干渉・接触）しているため、図 4 4 (D) に示すように、これら 2 つの組はシミュレーション時における距離測定の対象からは除外される。

【0 1 8 4】

次に、本発明の支援システム 1 0 を構成する 3 D モデルシミュレータ 3 0 の別の実施態様について、図 4 5 ～ 図 6 4 を参照しながら説明する。

図 4 5 は、本実施形態における別の実施態様の 3 D モデルシミュレータ 3 0 によるシミュレーション処理全体の処理の流れを説明するためのフローチャートであり、この図 4 5 に示すように、ここでは、シミュレータ 3 0 と組込みソフトウェア開発部 4 0 との同期処理（ステップ S 1 0 0 1 ），モータ処理（ステップ S 1 0 0 2 ），関節不良動作処理（ステップ S 1 0 0 3 ），センサ処理（ステップ S 1 0 0 4 ）および組込みソフトウェア開発部 4 0 の実行時間  $\Delta T (= T_S)$  の決定処理（ステップ S 1 0 0 5 ）が繰り返し実行される。

【0 1 8 5】

シミュレータ 3 0 と組込みソフトウェア開発部 4 0 との同期処理（ステップ S 1 0 0 1 ）については、既に図 1 2 ～ 図 1 4 ， 図 2 0 や 図 3 2 ～ 図 3 4 により前述した通りであるため、その説明は省略する。それ以外の処理について、以下順次説明する。

図 4 6 は、モータ処理ルーチンを説明するためのフローチャートである。この図 4 6 に示すフローチャートは、基本的には、前述した実施形態における、図 2 1 の一点鎖線で囲った部分からなるモータ処理にモータの故障の概念を取り入れたものである。

【0 1 8 6】

図 4 6 に示すように、ステップ S 1 1 0 1 では、最短距離 D を求めるために先ずその最短距離 D に  $\infty$  が初期入力され、ステップ S 1 1 0 2 では、未処理のモータ

タが存在しているか否かが判定され、未処理のモータが存在する場合（YESルート）、未処理のモータ1つずつについてステップS1103～S1110の処理が行なわれる。

【0187】

即ち、まず、未処理のモータそれぞれについて、そのモータが故障しているか否かが判定される（ステップS1103）。モータの故障の有無は、ユーザによりあらかじめ設定され、フラグとして保存されている。

故障したモータについては処理を行わず（ステップS1103のYESルートからステップS1102）、故障していないモータについては（ステップS1103のNOルート）、組込みソフトウェア開発部40側から送られてきた、今処理しようとしているモータに関するアクチュエータ指令信号に応じて、そのモータの姿勢変位量 $\Delta\theta$ が決定される（ステップS1104）。

【0188】

そして、 $\Delta\theta = 0$ であるか否かを判定し、 $\Delta\theta = 0$ である場合（YESルート）には、ステップS1102に戻る一方、 $\Delta\theta = 0$ でなければ（NOルート）、そのモータのそれまでの姿勢 $\theta$ に今回の変位量 $\Delta\theta$ が加算され（ステップS1106）、そのモータリンクの移動に伴って移動するリンクの姿勢変位量を計算するリンク姿勢移動サブルーチン（図47を参照しながら後述）が実行される（ステップS1107）。

【0189】

その後、部品間最短距離 $d$ が求められ（ステップS1108）、 $d < D$ か否かが判定され（ステップS1109）、 $d < D$ でないとき（NOルート）はステップS1102に戻る一方、 $d < D$ のとき（YESルート）、すなわち、今回求めた最短距離 $d$ がこれまでに求めた最短距離 $D$ よりも短いときは、最短距離 $D$ を今回の求めた最短距離 $d$ に置き換えて、最短距離 $D$ を更新する（ステップS1110）。

【0190】

そして、全てのモータに関し、今回の演算時間 $\Delta T$ の間の姿勢変更が終了すると、このモータ処理を終了する。

図 4 7 は、図 4 6 に示すモータ処理ルーチン（ステップ S 1 1 0 7）で実行されるリンク姿勢移動サブルーチンを説明するためのフローチャートである。

この図 4 7 に示すように、リンク姿勢移動サブルーチンでは、まず、リンク姿勢移動（図 4 6 のステップ S 1 1 0 7 でコールされたときは、モータリンクの姿勢移動）を受けて（ステップ S 1 2 0 1）、その姿勢移動のあったリンクに対する受動リンクに未計算の受動リンクが存在するか否かを判定する（ステップ S 1 2 0 2）。

【0 1 9 1】

未計算の受動リンクが存在する場合（YES ルート）、今回姿勢移動のあったリンクとそのリンクに対する受動リンクとの間にクラッチが存在するか否かを判定し（ステップ S 1 2 0 3）、クラッチが存在する場合（YES ルート）、そのクラッチ部品の姿勢がオンの位置にあるか否かを判定する（ステップ S 1 2 0 4）。クラッチ部品の存在や、クラッチ部品の姿勢とクラッチのオン、オフとの関係は、ユーザにより予め定義される。

【0 1 9 2】

クラッチ部品が存在しない場合（ステップ S 1 2 0 3 の NO ルート）、あるいは、クラッチ部品が存在していてもオン状態にある場合（ステップ S 1 2 0 4 の YES ルート）、ステップ S 1 2 0 5 に進み、ギア、カム関係に基づいて受動部品の移動量を計算し、さらにステップ S 1 2 0 6 において、今回移動量を計算した受動部品を能動部品としたときのその能動部品に対する受動部品の移動量を求めるために、再度、このリンク姿勢移動サブルーチンがコールされ、そのサブルーチンを実行してからステップ S 1 2 0 2 へ戻る。なお、ステップ S 1 2 0 4 でクラッチ部品がオン状態ではないと判定された場合（NO ルート）、直接、ステップ S 1 2 0 2 に戻る。

【0 1 9 3】

このようにして、リンク姿勢移動サブルーチンは、図 4 1 に示す検索ルーチンと同様、再帰的に実行され、この再帰的な実行により、1 つのモータを動かしたときに他の部品の移動量が高速に求められる。

図 4 8 は、図 4 5 のステップ S 1 0 0 3 で実行される関節不良動作処理ルーチ



ンを説明するためのフローチャートである。

【0194】

この図48に示すように、関節不良動作不良ルーチンでは、まず、3次元機構モデルを画面表示した後（ステップS1301）、ユーザが新たに不良リンクを指定したか否か（ステップS1302）、未処理のユーザ指定不良リンクが存在するか否か（ステップS1305）、未処理の、干渉が起きたリンクが存在するか否か（ステップS1310）、および不良リンクが存在するか否か（ステップS1314）が、順次、判定される。

【0195】

ステップS1302において、ユーザが新たに不良リンクを指定したと判定されると（YESルート）、ステップS1303に進み、その新たに指定された不良リンクを駆動しているモータを検索し、そのモータとそのモータの移動方向とをリストに追加してから（ステップS1304）、ステップS1305へ進む。

ステップS1305において未処理のユーザ指定不良リンクが存在すると判定されると（YESルート）、ステップS1306に進み、その不良リンクを駆動しているモータの移動方向が逆転したか否かが判定され、逆転していない場合（NOルート）はステップS1307に進んでそのモータの姿勢を旧姿勢に戻し、そのモータの姿勢が旧姿勢に戻されたことに対応してリンクの姿勢を旧姿勢に戻すために、図47に示すリンク姿勢移動サブルーチンがコールされて実行された後（ステップS1308）、ステップS1305に戻る。一方、ステップS1306においてそのモータの移動方向が逆転したと判定されると（YESルート）、ステップS1309に進み、そのモータを不良リンクのリストから削除してから、ステップS1305に戻る。

【0196】

これにより、不良リンクが、ある一方向に移動しようとしたときに何らかの原因で引っ掛かって動かなくなり、その不良リンクが一旦逆方向に移動することによりその引っ掛かりが解除されて正常に戻るという動作がシミュレートされる。

ステップS1310において、未処理の、干渉が起きたリンクが存在する旨判定されると（YESルート）、ステップS1311に進み、その不良リンクを駆

動しているモータの検索が行なわれ、そのモータの姿勢が旧姿勢に戻され（ステップ S 1 3 1 2）、そのモータによって駆動される他のリンクの姿勢を旧姿勢に戻すべく、図 4 7 に示すリンク機構サブルーチンがコールされて実行された後（ステップ S 1 3 1 3）、ステップ S 1 3 1 0 に戻る。

【0 1 9 7】

これにより、あるリンクが、他のリンクと干渉してそれ以上動かすことのできない状態をシミュレートすることができる。

ステップ S 1 3 1 4 において不良リンクが存在すると判定されると（YES ルート）、その 3 次元機構モデルを画像上に再描画する（ステップ S 1 3 1 5）。これにより、ステップ S 1 3 0 1 における描画と合わせて、3 次元機構モデルの再描画結果をシミュレーション画面上で見ると、不良リンクを受動部品に持ったモータおよびそのモータにより駆動される全リンクの動きが、不自然な状態（例えば、カクカクして、いかにもひっかかった様子）で表示される。

【0 1 9 8】

なお、ここに示した例では、モータの姿勢を元に戻すことによって関節不良の状態を実現したが、関節値の上限あるいは下限を一時的に現在値に設定することで関節不良の状態を実現することも可能である。

図 4 9 は、図 4 5 のステップ S 1 0 0 4 で実行されるセンサ処理ルーチンを説明するためのフローチャートである。

【0 1 9 9】

前述の実施形態では、センサはオン、オフの信号を出力するオン／オフセンサとしていたが、ここでは、センサとして、オン／オフセンサのほか、ポテンシオメータやエンコーダも存在し、さらにセンサの不良も考慮されている。

図 4 9 に示すように、まず、ステップ S 1 4 0 1 で、未処理のセンサの有無が判定され、未処理のセンサが存在すると（YES ルート）、ステップ S 1 4 0 2 に進み、その未処理のセンサがオン／オフセンサであるか否かが判定される。オン／オフセンサであった場合（YES ルート）、ステップ S 1 4 0 3 において、オン／オフセンサ処理が行なわれ、そのオン／オフセンサを処理した結果のセンサ値（オンあるいはオフ）が出力され、その後、ステップ S 1 4 0 1 に戻る。

## 【 0 2 0 0 】

ステップ S 1 4 0 2 において、その未処理のセンサがオン／オフセンサではないと判定されると（N O ルート）、ステップ S 1 4 0 4 に進み、そのセンサがポテンシオメータであるか否かが判定され、ポテンシオメータであると判定されると（Y E S ルート）、ステップ S 1 4 0 5 に進んでポテンシオメータ前処理が行なわれた後、ステップ S 1 4 0 6 でエンコーダ処理が行なわれその結果としてのセンサ値を出力してから、ステップ S 1 4 0 1 に戻る。

## 【 0 2 0 1 】

一方、ポテンシオメータではないと判定された場合（N O ルート）、つまり、未処理のセンサがエンコーダであると判定された場合、直接、ステップ S 1 4 0 6 のエンコーダ処理に進み、その結果としてのセンサ値を出力してから、ステップ S 1 4 0 1 に戻る。

図 5 0 は、図 4 9 のステップ S 1 4 0 3 で実行されるオン／オフセンサ処理ルーチンを説明するためのフローチャートである。

## 【 0 2 0 2 】

このオン／オフセンサ処理ルーチンでは、そのオン／オフセンサの故障もシミュレートされており、図 5 0 に示すように、まず、ステップ S 1 5 0 1 でそのオン／オフセンサがオンの状態のまま故障しているか否かが判定され、ステップ S 1 5 0 3 でそのオン／オフセンサがオフの状態のまま故障しているかが判定される。

## 【 0 2 0 3 】

オン状態のまま故障している場合（ステップ S 1 5 0 1 の Y E S ルート）、出力予定値としてオンが設定され（ステップ S 1 5 0 2 ）、オフ状態のまま故障している場合（ステップ S 1 5 0 3 の Y E S ルート）、出力予定値としてオフが設定され（ステップ S 1 5 0 4 ）、ステップ S 1 5 0 8 に進んでその出力予定値がセンサ値として出力される。そのオン／オフセンサがオン状態あるいはオフ状態で故障しているか否かはユーザにより予め設定される。

## 【 0 2 0 4 】

そのオン／オフセンサが故障していない場合（ステップ S 1 5 0 1 , S 1 5 0

3のNOルート)、ステップS1505に進み、そのセンサリンクの姿勢から出力予定値(オンあるいはオフ)が決定され(ステップS1505)、次いで、性能劣化による出力予定値変更処理(ステップS1506)、および、チャタリングによる出力予定値変更処理(ステップS1507)を実行してから、ステップS1508へ進む。

#### 【0205】

図51は、図50のオン/オフセンサ処理ルーチンのステップS1506で実行されるセンサ性能劣化処理ルーチンを説明するためのフローチャートである。このルーチンでは、オン/オフセンサが故障(出力がオンまたはオフに固定)までいかなくても、応答性能が劣化し、そのオン/オフセンサの出力をオンまたはオフに変化させる状態が生じた後、暫らくは前の状態を保ち、遅れてオンまたはオフに変化する状態がシミュレートされる。

#### 【0206】

図51に示すように、まず、ステップS1601で、旧出力予定値 $d$ と出力予定値とが等しいか否かが判定され、変化していた場合(NOルート)、ステップS1602に進み出力予定値がオンかオフかが判定される。出力予定値がオンの場合(YESルート)、ステップS1603に進み、センサ出力がオフからオンに変化するときの遅れ時間を $T_d$ に設定してから、ステップS1605へ進む。一方、出力予定値がオフの場合(ステップS1602のNOルート)、ステップS1604に進み、センサ出力がオンからオフに変化するときの遅れ時間を $T_d$ に設定してから、ステップS1605に進む。

#### 【0207】

ステップS1601で旧出力予定値 $d$ と出力予定値とが等しいと判定された場合(YESルート)、あるいは、ステップS1603、S1604の処理の完了後、ステップS1605で $T_d > 0$ か否かが判定され、 $T_d > 0$ の場合(YESルート)、ステップS1606に進んで $T_d$ から組込みソフトウェア開発部40の実行時間 $\Delta T$ が減算され、ステップS1607において出力予定値を反転してから(出力予定値がオンのときはオフ、出力予定値がオフのときはオン)、ステップS1608に進む。

## 【0208】

ステップS1605で $T_d > 0$ でhないと判定された場合（NOルート）、あるいは、ステップS1607の処理の完了後、ステップS1608では、旧出力予定値dに今回の出力予定値が格納される。

図45に示すように、センサ処理S1004には繰り返し実行され、図51に示すセンサ性能劣化処理も繰り返し実行され、 $T_d > 0$ のときは、この処理が繰り返される度に、 $T_d$ から $\Delta T$ が減算され $T_d \leq 0$ に達するとセンサが変化することになる。

## 【0209】

図52は、図50のオン／オフセンサ処理ルーチンのステップS1507で実行される、チャタリングによる出力予定値変更処理ルーチンを説明するためのフローチャートである。このルーチンではオン／オフセンサがオンからオフ、あるいはオフからオンに変化するときのチャタリングがシミュレートされる。

この図52に示すように、まず、ステップS1701で、旧出力予定値cと出力予定値とが等しいか否かが判定され、変化していた場合（NOルート）、ステップS1702に進み、出力予定値がオンかオフかが判定される。出力予定値がオンの場合（YESルート）、ステップS1703に進み、センサ出力がオフからオンに変化するときのチャタリング時間を $T_c$ に設定してから、ステップS1705に進む。一方、出力設定値がオフの場合（ステップS1702のNOルート）、ステップS1704に進み、センサ出力がオンからオフに変化するときのチャタリング時間を $T_c$ に設定してから、ステップS1705に進む。

## 【0210】

そして、ステップS1705では、旧出力予定値cに出力予定値が設定され、この処理を終了した後、もしくは、ステップS1701で旧出力予定値cと出力予定値とが等しいと判定された場合（YESルート）、ステップS1706で $T_c > 0$ か否かが判定される。

そして、 $T_c > 0$ の場合（YESルート）、ステップS1707に進んで $T_c$ から組込みソフトウェア開発部40の実行時間 $\Delta T$ が減算され、ステップS1708において、0～99の値をとる乱数が発生されてその発生した乱数が予め設定

された発生確率（0～99のうちのいずれかの値）と比較される。その結果、乱数＞発生確率の場合（YESルート）、出力予定値のオン、オフを反転してから（ステップS1709）、処理を終了する。一方、ステップS1708で乱数＞発生確率でないと判定された場合（NORルート）や、ステップS1706で $T_c > 0$ ではないと判定された場合（NORルート）は、そのまま、処理を終了する。

#### 【0211】

このようにして、 $T_c > 0$ の間、確率的にオン、オフが変化するチャタリングがシミュレートされる。

次に、図49のステップS1406で実行されるエンコーダ処理について説明する。

図53は、エンコーダの変位（角度等）に対する出力波形の一例を示す図、図54（A）～（E）は、エンコーダの、各種の出力波形を示す図である。

#### 【0212】

ここでは、図53に示すように、エンコーダの出力波形が、波長 $L$ の周期的な関数 $f(\theta)$ である場合について考える。ただし、関数 $f(\theta)$ は、図54（A）～図54（E）に示すような各種の波形をとることができる。ここで、波長 $L$ は関数 $f(\theta)$ の一周期に対応するもので、例えば、このエンコーダがロータリ式のエンコーダであって回転角度1.0度ごとに一周期の信号を出力するエンコーダであるときは、 $L = 1.0$ （度）となる。

#### 【0213】

ここで、変位 $\theta$ を電気角 $\phi = 2\pi\theta/L$ に変換した場合、関数 $f(\theta)$ が関数 $g(\phi)$ として表記されるものとして、以下、エンコーダ機能の原理を説明する。

前回行なったシミュレーション時のエンコーダの電気角を $\phi_0$ 、今回、すなわち前回行なったシミュレーションから組込みソフトウェア開発部40の演算時間 $\Delta T$ だけ進んだ時刻の電気角を $\phi$ 、エンコーダの電気角速度を $\omega$ とすると、前回から今回までの間のシミュレーションサイクルにおけるエンコーダの移動量は $\omega \Delta T$ となり、

$$\phi = \phi_0 + \omega \Delta T$$

が成立する。

【0214】

しかし、エンコーダはパルス数を測定するため、今回のシミュレーションサイクルにおける $\omega \Delta T$ が $2\pi$ よりも大きい場合、単純に $g(\phi)$ を出力するわけにはいかず、所定のパルス数のパルスを出力した後に $g(\phi)$ を出力する必要がある。

図55は、エンコーダの出力処理ルーチンを説明するためのフローチャートである。

【0215】

ここでは、図55に示すように、まず、前回のシミュレーション終了時のエンコーダ出力 $g(\phi_0)$ を出力し（ステップS1801）、今回のシミュレーション終了時の電気角 $\phi$ と前回の電気角 $\phi_0$ との差分 $\phi - \phi_0$ が、 $\phi - \phi_0 > \pi/2$ であるか否か（ステップS1802）、あるいは、 $\phi_0 - \phi > \pi/2$ であるか否か（ステップS1804）が判定される。

【0216】

$\phi - \phi_0 > \pi/2$ の場合（ステップS1802のYESルート）、ステップS1803に進んで、 $\phi_0$ に $\pi/2$ が加算され、ステップS1806において50 $\mu$ sec待機した後、ステップS1801に戻って新たな $\phi_0$ に対する $g(\phi_0)$ が出力される。 $\phi_0 - \phi > \pi/2$ の場合（ステップS1804のYESルート）、ステップS1805に進み $\phi_0$ から $\pi/2$ が減算され、その後は同様にステップS1806において50 $\mu$ sec待機した後、ステップS1801に戻って新たな $\phi_0$ に対する $g(\phi_0)$ が出力される。

【0217】

つまり、ここでは $|\phi - \phi_0| > \pi/2$ のときは、 $|\phi - \phi_0| \leq \pi/2$ になるまで、5kHzの周波数を上限として（ステップS1806）、 $\phi$ が $\pi/2$ ずつ変化したときの離散的な周期信号が出力される。

$|\phi - \phi_0| \leq \pi/2$ に達すると（ステップS1802、S1804のNOルート）、ステップS1807に進み、今回のシミュレーションにおけるエンコーダの最終値である $g(\phi)$ が出力され、次のシミュレーションのために今回の

シミュレーションにおける  $\phi$  が新たな  $\phi_0$  に置き換えられる（ステップ S 1 8 0 8）。

#### 【 0 2 1 8 】

このエンコーダ信号は、組込みソフトウェア開発部 4 0 とは独立して動作するカウンタ回路によって受信されることを想定しており、したがってエンコーダ信号発信時に組込みソフトウェア開発部 4 0 が停止していても問題はない。

なお、図 5 5 に示すフローチャートは、パルス信号を  $g(\phi)$  の 1 つのみ出力するタイプのエンコーダを対象として表現したものであるが、 $g(\phi)$  の他に  $g(\phi + \pi/2)$  を出力することにより、A 相、B 相の二相信号を出力するタイプのエンコーダも表現することができる。

#### 【 0 2 1 9 】

また、 $g(\phi)$  のほかに  $\phi - \phi_0$  の正負の符号をオン/オフで出力することにより、アップ/ダウン信号を出力するタイプのエンコーダも表現することができる。

ここで、上記のシミュレーションに先立って、パルスの波形〔図 5 4 (A) ～ 図 5 4 (E) 参照〕や、分解能や、電気角  $\phi$  がゼロになる位置をあらわすオフセットを入力することにより、エンコーダの特性がユーザにより定義される。

#### 【 0 2 2 0 】

図 5 6 は、図 4 9 に示す処理ルーチンのステップ S 1 4 0 6 で実行されるエンコーダ処理ルーチンを説明するためのフローチャートである。

ここでは、エンコーダの故障についても考慮されており、エンコーダに故障があったときの 3 次元機構モデルのシミュレーションのために、予めユーザにより以下の故障フラグ 0 ～ 6 が設定される。

#### 【 0 2 2 1 】

故障フラグ 0 : 正常

故障フラグ 1 : A 相が  $V_{\max}$  で故障

故障フラグ 2 : A 相が  $V_{\min}$  で故障

故障フラグ 3 : B 相が  $V_{\max}$  で故障

故障フラグ 4 : B 相が  $V_{\min}$  で故障



故障フラグ 5 : アップ/ダウン信号故障 (High に固定)

故障フラグ 6 : アップ/ダウン信号故障 (Low に固定)

図 5 6 に示すように、まず、ステップ S 1 9 0 1 で、変位  $\theta$  が電気角  $\phi$  に変換され、次いで、故障フラグが 1 であるか否か (ステップ S 1 9 0 2)、2 であるか否か (ステップ S 1 9 0 3) が判定され、それらの何れでもないときは、ステップ S 1 9 0 4 に進んで、図 5 5 に示す  $g(\phi)$  出力処理ルーチンが実行される。

#### 【0 2 2 2】

一方、ステップ S 1 9 0 2 において故障フラグが 1 であると判定されるとステップ S 1 9 0 5 に進み、 $V_{\max}$  で故障していることに対応して  $g(\pi/4)$  が出力される一方、ステップ S 1 9 0 3 において故障フラグが 2 であると判定されるとステップ S 1 9 0 6 に進み、 $V_{\min}$  で故障していることに対応して  $g(3\pi/4)$  が出力される。

#### 【0 2 2 3】

つづいて、ステップ S 1 9 0 7 では、このエンコーダが 2 相出力タイプのエンコーダであるか否か、ステップ S 1 9 0 8 では、このエンコーダがアップ/ダウン信号を出力するタイプのエンコーダであるか否かが判定され、いずれでもない場合、つまり単純に  $g(\phi)$  のみを出力するタイプのエンコーダのときはそのままこのルーチンを抜ける。

#### 【0 2 2 4】

ステップ S 1 9 0 7 においてこのエンコーダが 2 相出力タイプのエンコーダであると判定されると、ステップ S 1 9 0 9 において故障フラグが 3 であるか否か、ステップ S 1 9 1 0 において故障フラグが 4 であるか否かが判定される。それらのいずれでもなかったときは、ステップ S 1 9 1 1 に進み、図 5 5 における  $\phi$  を  $\phi + \pi/2$  に置き換えた出力処理ルーチンが実行される。

#### 【0 2 2 5】

ステップ S 1 9 0 9 において故障フラグが 3 であると判定されるとステップ S 1 9 1 2 に進み、B 相が  $V_{\max}$  で故障していることに対応して  $g(\pi/4)$  が出力され、これと同様に、ステップ S 1 9 1 0 において故障フラグが 4 であると判

定されるとステップ S 1 9 1 3 に進み、B 相が  $V_{\min}$  で故障していることに対応して  $g(3\pi/4)$  が出力される。

【0 2 2 6】

また、ステップ S 1 9 0 8 においてこのエンコーダがアップ/ダウン (U/D) 信号出力タイプのエンコーダであると判定されると、ステップ S 1 9 1 4 に進み故障フラグが 5 であるか否かが判定され、ステップ S 1 9 1 5 では故障フラグが 6 であるか否かが判定される。

それらのいずれでもないときはステップ S 1 9 1 6 に進みそのエンコーダの移動方向が判定され、アップ方向 (U ルート) のときはステップ S 1 9 1 7 に進んで 'H i g h' が出力され、ダウン方向 (D ルート) のときはステップ S 1 9 1 8 に進んで 'L o w' が出力される。

【0 2 2 7】

一方、ステップ S 1 9 1 4 において故障フラグが 5 であると判定されるとステップ S 1 9 1 7 に進み、エンコーダの移動方向のいかんによらず 'H i g h' が出力され、これと同様に、ステップ S 1 9 1 5 において故障フラグが 6 であると判定されるとステップ S 1 9 1 8 に進み、エンコーダの移動方向のいかんによらず 'L o w' が出力される。

【0 2 2 8】

このようにしてエンコーダを表現することができるとともに、エンコーダの故障を表現することができる。ただし、エンコーダの場合、オン/オフセンサの場合と異なり、処理に入った直後に故障中を示すのではなく、センサ信号を出力する際に故障フラグの判定や処理が行なわれる。

図 5 7 は、ポテンシオメータの関数値の一例を説明するための図である。ここでは、ポテンシオメータが、図 5 4 (D) に示す三角波を出力する、分解能の極端に低いエンコーダであるとして実現している。例えば図 5 3 に示す波長  $L$  として 3 6 0 度を設定すれば、図 5 7 に示すように、 $\theta_{\max} - \theta_{\min} = 1 8 0$  度の変位で出力電圧  $V$  を最小値  $V_{\min}$  から最大値  $V_{\max}$  まで変換するポテンシオメータを表現することができる。

【0 2 2 9】

図 5 8 は、図 4 9 に示すセンサ処理ルーチンのステップ S 1 4 0 5 で実行されるポテンシオメータ前処理ルーチンを説明するためのフローチャートである。

ポテンシオメータの場合、図 5 7 に実線で示すように、 $\theta_{\min}$ 以下での $\theta$ では出力電圧 $V$ は最低値に固定され、 $\theta_{\max}$ 以上の $\theta$ では出力電圧 $V$ は最大値に固定されることがあり、この図 5 8 ではそのようなポテンシオメータを実現している。

#### 【0 2 3 0】

即ち、図 5 8 に示すように、ステップ S 2 0 0 1 において $\theta > \theta_{\max}$ を満足するか否かが判定され、それを満足するときはステップ S 2 0 0 2 に進んで $\theta$ が $\theta_{\max}$ に固定される。

また、これと同様に、ステップ S 2 0 0 3 において $\theta < \theta_{\min}$ を満足するか否かが判定され $\theta < \theta_{\min}$ を満足するときはステップ S 2 0 0 4 に進んで $\theta$ が $\theta_{\min}$ に固定される。

#### 【0 2 3 1】

ポテンシオメータの場合、このような前処理が行なわれた後、図 5 6 に示すエンコーダ処理が実行される（図 4 9 参照）。

このタイプのポテンシオメータの場合、ユーザは、 $\theta_{\min}$ 、 $\theta_{\max}$ および、 $V_{\min}$ 、 $V_{\max}$ を入力すると、システムは、その入力値を自動的に三角波 $f(\theta)$ の波長、振幅、オフセットに変換する。

#### 【0 2 3 2】

なお、ストッパがないポテンシオメータなど、出力が周期的に変化するポテンシオメータの場合は、分解能が低い三角波や鋸波を出力するエンコーダとして取り扱われる。

図 5 9 は、図 4 5 のステップ S 1 0 0 5 において実行される $\Delta T$ 決定処理ルーチン（シミュレーション間隔決定処理ルーチン）を説明するためのフローチャートである。

#### 【0 2 3 3】

この図 5 9 に示すように、ここでは、まず、ステップ S 2 1 0 1 において、性能劣化（応答遅れあるいはチャタリング）を起こしたセンサが存在するか否かが

判定され、そのようなセンサが存在しないときは、図46に示す処理で求めた部品間の最短距離Dに応じた $\Delta T$ が決定される（ステップS2102）。一方、性能劣化を起こしたセンサが存在するときは、 $\Delta T$ としてとり得る値の最小値が設定される（ステップS2103）。

#### 【0234】

ここで、性能劣化を起こしたセンサが存在するときに $\Delta T$ を最小値に設定するのは、性能劣化の表現精度を高め、正確なシミュレーションを行なうためである。なお、正常なセンサの場合、遅れ時間はゼロに設定されているため、問題なく動作する。

図60（A）および図60（B）は、干渉チェックによる関節可動範囲の検索法を説明するための図である。

#### 【0235】

まず、ユーザによって、どの方向の制限値を設定したいのかということと、どの部品とどの部品とが接触することにより制限が起きるのかということが選択され、その後、関節可動範囲の検索が行なわれる。

この関節可動範囲の検索においては、関節の姿勢 $x$ を初期値 $x_0$ から $\Delta x$ ずつ増加させ、その度に干渉チェックを行なう。すると、図60（A）に示すように、 $x$ が $x_0 + n \Delta x$ では干渉が発生せず、次の $x_0 + (n+1) \Delta x$ で干渉が起こる位置が判明する。

#### 【0236】

続いて、例えば $\Delta x' = \Delta x / 10$ などのように $\Delta x$ よりも小さい移動量 $\Delta x'$ を用いて、図60（B）に示すように、 $x$ を $x_0 + \Delta x$ から $\Delta x'$ ずつ増加させることにより、 $x$ が $x_0 + n \Delta x + n' \Delta x'$ では干渉が発生せず、 $x_0 + n \Delta x + (n' + 1) \Delta x'$ で干渉が発生する位置が判明する。

このような処理を数回繰り返すことにより、高速に且つ所望の精度で $x$ の制限値 $x_0 + n \Delta x + n' \Delta x' + n'' \Delta x'' \dots$ を求めることができる。ただし、ユーザが指定した二つの部品が、関節をどのように動かしていても干渉しない位置関係であることも考えられるので、一回目の干渉位置検索の際には検索数 $n$ に上限を設け、無限ループを防ぐことが好ましい。

## 【 0 2 3 7 】

図 6 1 (A) ~ 図 6 1 (C) は、干渉チェックによるカム関係の検索方法を説明するための図、図 6 2 は、干渉チェックによるカム関係検索ルーチンを説明するためのフローチャートである。

ここでは、例えば図 6 1 (A) に示すごとく、予め、ユーザにより、駆動部品 P 1、受動部品 P 2、駆動部品 P 1 の初期姿勢、受動部品 P 2 の初期姿勢および駆動部品 P 1 の最終姿勢が指定される。

## 【 0 2 3 8 】

図 6 2 に示すルーチンでは、まず、 $\phi$ 、 $\psi$  に、それぞれ、駆動部品 P 1 の初期姿勢および受動部品 P 2 の初期姿勢が設定され (ステップ S 2 2 0 1)、次いで、受動部品 P 2 の非接触姿勢が検索されて  $\psi$  に格納される (ステップ S 2 2 0 2)。

この後、ステップ S 2 2 0 3 では、今回求めた、駆動部品 P 1 の姿勢  $\phi$  と、受動部品 P 2 の非接触姿勢  $\psi$  とのペアが登録され、例えば図 6 1 (B) に示すごとく駆動部品 P 1 の姿勢  $\phi$  が  $\Delta \phi$  だけ移動される (ステップ S 2 2 0 4)。

## 【 0 2 3 9 】

そして、駆動部品 P 1 の姿勢  $\phi$  が最終姿勢になるまで (ステップ S 2 2 0 5 で Y E S 判定となるまで)、 $\Delta \phi$  ずつ移動された駆動部品 P 1 に対する受動部品 P 2 の非接触姿勢  $\psi$  [図 6 1 (C) 参照] を検索・演算する処理 (ステップ S 2 2 0 2) と、姿勢  $\phi$ 、 $\psi$  の登録処理 (ステップ S 2 2 0 3) と、駆動部品 P 1 の移動処理 (ステップ S 2 2 0 4) とが繰り返される。

## 【 0 2 4 0 】

図 6 3 は、図 6 2 に示すカム関係検索ルーチンのステップ S 2 2 0 2 で実行される受動部品 P 2 の非接触姿勢検索ルーチンを説明するためのフローチャートである。

この図 6 3 に示すように、まず、ステップ S 2 3 0 1 において、受動部品 P 2 の現在の姿勢  $\psi$  が  $\psi_0$  に設定され、カウンタによるカウント値  $n$  に初期値ゼロが設定される。

## 【 0 2 4 1 】

ステップ S 2 3 0 2 では駆動部品 P 1 と受動部品 P 2 とが干渉しているか否かが判定され、干渉していないときは、現在の姿勢  $\psi$  が非接触姿勢となる（ステップ S 2 3 1 5）。

ステップ S 2 3 0 2 で干渉していると判定されると、ステップ S 2 3 0 3 においてカウント値  $n$  が 1 だけインクリメントされ、受動部品 P 2 の姿勢  $\psi$  として  $\psi_0 + n \Delta \psi$  が設定され（ステップ S 2 3 0 4）、再度干渉しているか否かが判定される（ステップ S 2 3 0 5）。まだ干渉していたときは、今度は受動部品 P 2 の姿勢  $\psi$  として  $\psi_0 - n \Delta \psi$  が設定され（ステップ S 2 3 0 6）、干渉しているか否かがもう一度判定される（ステップ S 2 3 0 7）。まだ干渉していたときは、ステップ S 2 3 0 8 に進み、カウント値  $n$  が  $n$  の上限か否かが判定され、 $n$  の上限であったときは検索失敗とされる（ステップ S 2 3 0 9）。カウント値  $n$  がまだ上限に達していなかったときは、ステップ S 2 3 0 3 に戻りカウント値  $n$  が 1 だけインクリメントされ、以下同様にして干渉チェックが行なわれる。

#### 【 0 2 4 2 】

ステップ S 2 3 0 5 で干渉していないと判定されると、ステップ S 2 3 1 0 に進み、 $\psi$  が  $\Delta \psi$  だけデクリメントされ、 $\Delta \psi$  が適当な整数  $N$  で除算され、その除算結果が  $\Delta \psi$  として設定され、 $\Delta \psi$  が前よりも小さい値に設定される（ステップ S 2 3 1 1）。

次いで、 $\psi$  が、上記のようにして小さな値に設定された  $\Delta \psi$  だけインクリメントされながら（ステップ S 2 3 1 3）、干渉しているか否かが判定され（ステップ S 2 3 1 2）、干渉した状態から干渉していない状態に変化すると、ステップ S 2 3 1 4 に進んで所定の精度まで検索したか否かが判定され、まだ精度が不十分なときはステップ S 2 3 1 0 に戻って  $\psi$  から  $\Delta \psi$  が減算され、ステップ S 2 3 1 1 において  $\Delta \psi$  としさらに小さい値が設定され、 $\psi$  をさらに小さい値である  $\Delta \psi$  ずつインクリメントしながら干渉チェックが行なわれる（ステップ S 2 3 1 2 , S 2 3 1 3）。

#### 【 0 2 4 3 】

このようにして所定の精度まで検索が行なわれると（ステップ S 2 3 1 4）、これにより非接触姿勢が定まる（ステップ S 2 3 1 5）。

ステップ S 2 3 0 7 において干渉していないと判定されたときも、ステップ S 2 3 1 0 ~ S 2 3 1 4 の処理と同様の処理が実行され、受動部品 P 2 の非接触姿勢が求められる。なお、ステップ S 2 3 0 7 において干渉していないと判定されたときは、ステップ S 2 3 1 0 に相当するステップにおいて  $\psi$  は  $\Delta \psi$  だけインクリメントされ、ステップ S 2 3 1 3 に相当するステップにおいて  $\psi$  は  $\Delta \psi$  だけデクリメントされる。

#### 【 0 2 4 4 】

図 6 4 は、図 6 2 に示すカム関係検索ルーチンのステップ S 2 2 0 2 で実行される受動部品の非接触姿勢検索ルーチンを説明するためのフローチャートである。この図 6 4 に示すルーチンは、受動部品に重力あるいはバネ付勢力が作用している場合に実行されるルーチンである。受動部品に作用する重力あるいはバネの付勢方向はユーザにより予め設定される。つまり、図 6 4 は、負方向に力が加わっている場合のフローチャートを示している。

#### 【 0 2 4 5 】

図 6 4 に示すルーチンでは、まず、ステップ S 2 4 0 1 において、受動部品の現在の姿勢  $\psi$  が  $\psi_0$  に設定され、カウンタのカウント値  $n$  に初期値ゼロが設定される。

ステップ S 2 4 0 2 では、駆動部品と受動部品とが現在干渉しているか否かが判定される。既に干渉しているときは、ステップ S 2 4 0 3 においてカウント値  $n$  を 1 だけインクリメントする。そして、この場合、駆動部品の今回の姿勢変化によって受動部品が干渉したものであるため、受動部品の姿勢を重力あるいはバネ付勢力に逆らって変化させ（ステップ S 2 4 0 4 ~ S 2 4 0 7）、干渉していない姿勢をみつけた後、 $\psi$  から  $\Delta \psi$  を減算することで（ステップ S 2 4 0 8）、一旦干渉した状態に戻す。ここで、ステップ S 2 4 0 3 ~ S 2 4 0 8 の処理は、それぞれ、図 6 3 のステップ S 2 3 0 3 ~ S 2 3 0 5, S 2 3 0 8, S 2 3 0 9, S 2 3 1 0 に対応している。

#### 【 0 2 4 6 】

一方、ステップ S 2 4 0 2 で干渉していないと判定されるときは、ステップ S 2 4 0 9 においてカウント値  $n$  を 1 だけインクリメントする。そして、この場合

、駆動部品の今回の姿勢変化によって受動部品が干渉していない状態に変化したものであるため、重力あるいはバネ付勢力の作用をシミュレートして受動部品が能動部品に干渉した状態を見つける（ステップS 2 4 1 0～S 2 4 1 3）。ここで、ステップS 2 4 0 9～S 2 4 1 3の処理は、それぞれ、図6 3のステップS 2 3 0 3，S 2 3 0 6～S 2 3 0 9に対応している。

【0 2 4 7】

そして、ステップS 2 4 0 8やS 2 4 1 1からステップS 2 4 1 4に進んだとき、 $\Delta\phi$ のきざみでぎりぎり干渉した状態にあり、 $\Delta\phi$ を適当な整数Nで除算しその除算結果を $\Delta\phi$ として設定することにより、 $\Delta\phi$ が前よりも小さい値に設定される。この後、 $\phi$ を $\Delta\phi$ ずつインクリメントしながら非干渉姿勢が検索される（ステップS 2 4 1 5，S 2 4 1 6）。

【0 2 4 8】

ステップS 2 4 1 7では所定の精度まで検索が行なわれたか否かが判定され、未だ精度が不十分なときはステップS 2 4 0 8に進んで $\phi$ から $\Delta\phi$ が減算されて干渉した状態に戻され、ステップS 2 4 1 4において $\Delta\phi$ が整数Nで除算されて $\Delta\phi$ としてさらに小さい値が設定されて検索が繰り返される。

このようにして所定の精度まで検索が行なわれると（ステップS 2 4 1 4）、非接触位置が決定される（ステップS 2 4 1 8）。ここで、ステップS 2 4 0 8，S 2 4 1 4～S 2 4 1 8の処理は、それぞれ、図6 3のステップS 2 3 1 0～S 2 3 1 5に対応している。

【0 2 4 9】

これにより、重力あるいはバネ付勢力が作用しない場合であっても作用する場合であっても、カム関係を決定することができる。

図6 5（A）～図6 5（C）は、駆動部材P 1側の溝P 1 0とその溝P 1 0に挿入されて駆動される受動部材P 2側のピンP 2 0との溝関係を設定する手法を説明するための図であり、図6 5（B）および図6 5（C）は、図6 5（A）の要部を拡大して示したものである。

【0 2 5 0】

図6 5（A）および図6 5（B）に示すような溝関係を設定しようとしたとき



、ピン P 2 0 は溝 P 1 0 に常に接触した状態にあるため、非干渉位置を求めることはできず、このような溝関係を、前述したカム関係の検索法を用いて求めることは困難である。

そこで、ここでは、図 6 5 (C) に示すように、ピン P 2 0 の中心に半径  $r$  の微小円筒 P 2 1 を一時的に取り付け、この微小円筒 P 2 1 と溝 P 1 0 との間の距離  $d$  を測定することで溝関係を求める。即ち、図 6 3 のステップ S 2 3 0 2, S 2 3 0 5, S 2 3 0 7, S 2 3 1 2 において、「干渉しているか?」の判定に代えて「 $d < R - r$  か?」の判定を行なうことにより、溝関係を検索することができる。

#### 【0 2 5 1】

さらに、以下に、本実施形態の 3 D モデルシミュレータ 3 0 における、各種機構（ローラ、ラチェット、モータ、センサ等）のシミュレーション手法について、図 6 6 ～図 8 2 を参照しながら説明する。

図 6 6 は、ギアやローラをリンクとして有する場合のリンク姿勢移動ルーチンを説明するためのフローチャートであり、この図 6 6 に示すルーチンは、例えば、前述したモータ処理ルーチン（図 4 6 のステップ S 1 1 0 7）で実行されるリンク姿勢移動サブルーチン（図 4 7 参照）に対応するものである。また、図 6 7 は関節値移動サブルーチンを説明するためのフローチャート、図 6 8 は関節値初期化サブルーチンを説明するためのフローチャートである。なお、以下では、リンクのことを関節と呼ぶ場合がある。

#### 【0 2 5 2】

ここで、ローラはギアの中の一機能であり、以下に説明するルーチンでは、図 6 6 ～図 6 8 に示すフローチャートのごとく、図 6 7 に示す関節値移動サブルーチンを再帰的に呼び出して実行することにより、現関節の動作に従って動作する従動関節（受動関節）を移動させていく。その移動途中において、従動関節の一つが制限値に到達した場合には、その関節値を予め設定した初期値に戻すために、図 6 8 に示す関節値初期化サブルーチンを再帰的に呼び出して実行している。

#### 【0 2 5 3】

図 6 6 に示すように、リンク姿勢移動（例えばモータリンクの姿勢移動）を受

けて本ルーチンが呼び出されると、モータリンクの移動に伴って現関節を移動させるべく、図 6 7 に示す関節値移動サブルーチンを呼び出して実行する（ステップ S 2 5 0 1）。そして、ステップ S 2 5 0 1 での関節値移動サブルーチンからの戻り値が“TRUE”か否かを判定し（ステップ S 2 5 0 2）、戻り値が“TRUE”である場合（YES ルート）には、現関節について、図 6 8 に示す関節値初期化サブルーチンを呼び出して実行してから（ステップ S 2 5 0 3）、本ルーチンの処理を終了する。ステップ S 2 5 0 2 で戻り値が“TRUE”でないと判定された場合（NO ルート）には、そのまま、本ルーチンの処理を終了する。

#### 【0 2 5 4】

関節値移動サブルーチンでは、図 6 7 に示すように、関節値の移動を行ってから（ステップ S 2 5 1 1）、その関節値が制限値に到達したか否かを判定する（ステップ S 2 5 1 2）。制限値に到達した場合（YES ルート）、戻り値として“TRUE”が出力される（ステップ S 2 5 1 3）。制限値に到達していない場合（NO ルート）、未処理の従動関節があるか否かを判定する（ステップ S 2 5 1 4）。未処理の従動関節がない場合（NO ルート）、戻り値として“FALSE”が出力される（ステップ S 2 5 1 5）。

#### 【0 2 5 5】

また、ステップ S 2 5 1 4 で未処理の従動関節があると判定された場合（YES ルート）、ギア、カム関係から従動関節の移動量を計算してから（ステップ S 2 5 1 6）、その従動関節について関節値移動サブルーチンを呼び出して実行し（ステップ S 2 5 1 7）、このステップ S 2 5 1 7 で実行されたサブルーチンからの戻り値が“TRUE”か否かを判定し（ステップ S 2 5 1 8）、戻り値が“TRUE”である場合（YES ルート）には、戻り値として“TRUE”が出力される（ステップ S 2 5 1 9）。戻り値が“TRUE”ではないと判定された場合（NO ルート）、ステップ S 2 5 1 4 に戻る。

#### 【0 2 5 6】

関節値初期化サブルーチンでは、図 6 8 に示すように、関節値として旧関節値を設定してから（ステップ S 2 5 2 0）、未処理の従動関節があるか否かを判定し（ステップ S 2 5 2 1）、未処理の従動関節がない場合（NO ルート）、この

関節値初期化サブルーチンを呼び出したステップの処理に戻る一方、未処理の従動関節がある場合（YESルート）、その従動関節について、関節値初期化サブルーチンを呼び出して実行してから（ステップS 2 5 2 2）、ステップS 2 5 2 1に戻る。

#### 【0 2 5 7】

図69（A）～図69（C）はギアやローラの動作について説明するための図である。

ここでは、図69（A）に示すように、駆動部品としてのドラムを回転させることにより、従動部品（受動部品）としての2つのスライダ1，2に駆動力が伝達され、これらのスライダ1，2が水平移動する機構について説明する。

#### 【0 2 5 8】

図69（A）に示す機構がギアで構成されている場合、図69（B）に示すように、ドラムの反時計回りの回転に伴いスライダ1の左端が左壁面に当たると（即ち、スライダ1が制限値に到達すると）、それ以上、ドラムやスライダ2は動かなくなる。

一方、図69（A）に示す機構において、ドラムとスライダ1との間のギアが滑りを許可するローラで構成されている場合、スライダ1が制限値に到達して動かなくなっても、図69（C）に示すように、ドラムは反時計回りに回転し続け、スライダ2は依然として左方向へ水平移動することになる。

#### 【0 2 5 9】

図70は、ローラをリンクとして有する場合の関節移動サブルーチンの変更部分を説明するためのフローチャートである。

このようなローラをシミュレータ30でシミュレートする場合、図67のステップS 2 5 1 8とステップS 2 5 1 9との間に、図70に示すように、従動関節がローラか否かを判定するステップS 2 5 3 0が追加され、ローラである場合（YESルート）、ステップS 2 5 1 4に戻る一方、ローラでない場合（NOルート）、ステップS 2 5 1 9へ進むようにする。つまり、処理対象の従動関節がローラかどうかを調べ、ローラである場合には、制限値処理を行なった後、その従動関節よりも上流側に制限値情報を流さないようにする。

## 【0260】

次に、ラチェットをシミュレータ30でシミュレートする手法について簡単に説明する。

ラチェットとは、自転車のフリーやラチェットレンチのように、ユーザが指定した方向には動力を伝達するが、その方向とは逆方向には動力伝達しないものである。

## 【0261】

このようなラチェットを、シミュレータ30で実現（シミュレート）する手法は、図47で説明したクラッチをシミュレートする手法とほぼ同様である。シミュレートすべき機構にラチェットが存在する場合、ラチェットを成す駆動部品の動く方向がユーザの指定した方向であれば動力を伝達させて、受動部品についての移動処理（関節値移動処理）を実行する一方、逆方向であれば動力を伝達させないようにする。

## 【0262】

図71（A）～図71（F）は、機構モデルとしての昇降機とその昇降機を成すリンクのツリー構造とを説明するための図、図72は、各リンクに持たせる、親子関係の切替情報を示す図、図73は、親切替チェックサブルーチンを説明するためのフローチャートである。

機構モデルを構成する各リンク（関節）は、一般に、ツリー状になっており、任意のリンクを動かすと、そのリンクの子（チャイルド）は全て同様に動くことになる。このようなデータ構造では、図71（A）～図71（F）に示すような昇降機を表現することが難しい。そこで、本実施形態では、クラッチと同様、何らかの条件（例えば第3の関節値）によって親を切り替える機能を追加する。

## 【0263】

図71（A）～図71（C）に示す昇降機は、水平に移動可能な車Cを、上下方向に移動可能なテーブルTにより、異なる高さの土台B、Bの間で移動させるものである。図71（A）に示す状態の昇降機は、図71（D）に示すように、車CおよびテーブルTを土台Bの子とするツリー構造で表される。また、図71（B）に示す状態の昇降機は、図71（E）に示すように、車CをテーブルTの

土台Bの子としテーブルTを土台Bの子とするツリー構造で表される。そして、図71(C)に示す状態の昇降機は、図71(F)に示すように、車CおよびテーブルTを土台Bの子とするツリー構造で表される。

#### 【0264】

このように親子関係が切り替わる機構（昇降機等）についてシミュレーションを行なう場合には、各リンク（関節）に図72に示すような親子関係の切替情報を予め持たせておき、シミュレータ30によるシミュレーションに際し、関節値が変化した場合（関節値移動後）、図72に示すような情報に基づいて、その変化に応じて親子関係に影響の出る可能性のある子部品について、親を再計算し、親の切替を行なう。

#### 【0265】

例えば、図67のステップS2511で関節値移動を行なった後、親子切替部品の有無を判定し、親子切替部品がある場合には、図73に示す親切替チェックサブルーチンを呼び出して実行する。

この図73に示すように、親切替チェックサブルーチンでは、まず、未処理の親部品候補が存在するか否かを判定し（ステップS2531）、存在しない場合（NORルート）、このサブルーチンの処理を終了する一方、存在する場合（YESルート）、未処理の、親が切り替わる条件が存在するか否かを判定する（ステップS2532）。

#### 【0266】

未処理の条件が存在する場合（ステップS2532のYESルート）、その条件が成り立つか否かを判定し（ステップS2533）、成立する場合（YESルート）、ステップS2532に戻る一方、成立しない場合（NORルート）、ステップS2531に戻る。そして、ステップS2532で未処理の条件が存在しないと判定された場合（NORルート）、親の切替を行なってから（ステップS2534）、このサブルーチンの処理を終了する。

#### 【0267】

次に、相対クラッチをシミュレータ30でシミュレートする手法について簡単に説明する。

クラッチが、何らかの条件（例えば第3の関節の関節値）によってギア、カム関係の動力伝達のオン／オフを決めるものであるに対し、相対クラッチとは、動力伝達のオン／オフを、任意の2つの部品の相対位置によって決めるもので、親子関係切替を用いた並べ替え機構などを実現する。

## 【0268】

一般のクラッチのシミュレーションに際して「クラッチ部品の姿勢はonの位置か」を調べていたのに対し（図47のステップS1204参照）、上述のような相対クラッチのシミュレーションに際しては、「部品Aから見て部品Bがonの位置か」を調べる。それ以外の処理は、一般のクラッチのシミュレーションと全く同様である。

## 【0269】

ここで、部品Aから見た部品Bの位置は、 $R_A^{-1}(P_A - P_B)$ により求めることができる。ただし、 $R_A$ は部品Aのローカル座標の絶対座標からの回転行列、 $P_A$ 、 $P_B$ はそれぞれ絶対座標から見た部品A、Bの位置である。

次に、DCモータをシミュレータ30でシミュレートする手法について、図74および図75を参照しながら説明する。なお、図74は、DCモータのシミュレーション手順を説明するためのフローチャート、図75は、DCモータの入力値とモータ指令値との対応関係の一例を示す図である。

## 【0270】

DCモータのシミュレーションを行なう際には、DCモータの入力値とモータ指令値との対応関係を、例えば図75に示すようなテーブルとして予め登録しておく。このテーブルは、図26において前述した対応表と同様のものである。ただし、図75に示すテーブルでは、DCモータの2つのポート（ポート番号11，12）に入力される値（アクチュエータ指令信号）と、目標速度(deg/sec)および整定時間(msec)との対応関係が設定されている。

## 【0271】

図75のテーブルを用いた場合、ポート番号11，12にそれぞれ0，1が入力されると、目標速度+100deg/secおよび整定時間500secがモータ指令値として出力され、ポート番号11，12にそれぞれ1，0が入力されると、目標

速度 - 1 0 0 deg/sec および整定時間 5 0 0 sec がモータ指令値として出力される。

#### 【 0 2 7 2 】

このようなテーブルを設定・登録しておいてから、図 7 4 に示す手順でシミュレータ 3 0 により DC モータのシミュレーションを行なう。即ち、組込みソフトウェア開発部 4 0 からのアクチュエータ指令信号を、ポート番号の状態入力として認識してから（ステップ S 2 5 4 1）、その入力値を図 7 5 のテーブルによりモータ指令値（目標速度および整定時間）に変換する（ステップ S 2 5 4 2）。

#### 【 0 2 7 3 】

そして、既に算出されている今回のシミュレーション時間  $T_S (= \Delta t)$  を入力し（ステップ S 2 5 4 3）、今回のシミュレーション時間  $T_S (= \Delta t)$  での DC モータの回転角を計算する（ステップ S 2 5 4 4）。このとき、回転角は、目標速度  $\times$  シミュレーション時間として算出できるが、モータ指令値が入力されてからのトータルのシミュレーション時間が整定時間以下であれば、前述したごとく、式（4）を用いて回転角が算出される。この後、その DC モータに連結した全ての関節（リンク）の関節値を計算して処理を終了する（ステップ S 2 5 4 5）。

#### 【 0 2 7 4 】

次に、ステッピングモータをシミュレータ 3 0 でシミュレートする手法について、図 7 6 ～ 図 7 8 を参照しながら説明する。なお、図 7 6 は、ステッピングモータのシミュレーション手順を説明するためのフローチャート、図 7 7 は、ステッピングモータの加速パターンを説明するための図、図 7 8 は、パルス数計算サブルーチンを説明するためのフローチャートである。

#### 【 0 2 7 5 】

ステッピングモータのシミュレーションを行なう際には、例えば図 7 7 に示すような、ステッピングモータの加速パターン  $T_h [N]$  を事前に登録しておく。この加速パターン  $T_h [N]$  としては、加速を開始して  $i$  番目のパルスの時間間隔が  $t_i$  として登録されている。

このような加速パターン  $T_h [N]$  を登録しておいてから、図 7 6 に示す手順

でシミュレータ 3 0 によりステッピングモータのシミュレーションを行なう。即ち、組込みソフトウェア開発部 4 0 から、アクチュエータ指令信号であるモータ指令値（目標速度）を入力するとともに（ステップ S 2 5 5 1）、既に算出されている今回のシミュレーション時間  $T_S (= \Delta t)$  を入力し（ステップ S 2 5 5 2）、今回のシミュレーション時間  $T_S (= \Delta t)$  内に、ステッピングモータに対して出力されるパルス数を、図 7 8 に示すパルス数計算サブルーチンにより求める（ステップ S 2 5 5 3）。

【 0 2 7 6 】

そして、ステップ S 2 5 5 3 で計算されたパルス数に基づいて、ステッピングモータの回転角を計算し（ステップ S 2 5 5 4）、そのステッピングモータに連結した全ての関節（リンク）の関節値を計算して処理を終了する（ステップ S 2 5 5 5）。

なお、パルス数計算サブルーチンでは、図 7 8 に示すように、まず、パルス数  $p$  として累積パルス数を、シミュレーション時間  $t$  として 0 を初期設定しておいてから（ステップ S 2 5 6 1）、 $t$  として  $t + T_h [p]$  を設定し（ステップ S 2 5 6 2）、置き換え後の時間  $t$  がシミュレーション時間  $T_S$  よりも小さいか否かを判定する（ステップ S 2 5 6 3）。 $t < T_S$  であれば（YES ルート）、 $p$  として  $p + 1$  を設定し（ステップ S 2 5 6 4）、ステップ S 2 5 6 2 に戻る一方、シミュレーション時間  $t$  が  $T_S$  に達すると、このパルス数計算サブルーチンを終了し、パルス数  $p$  を計算結果として出力する。

【 0 2 7 7 】

次に、AC モータをシミュレータ 3 0 でシミュレートする手法について、図 7 9 を参照しながら説明する。なお、図 7 9 は、AC モータのシミュレーション手順を説明するためのフローチャートである。

この図 7 9 に示すように、シミュレータ 3 0 によりステッピングモータのシミュレーションを行なう際には、まず、組込みソフトウェア開発部 4 0 から、アクチュエータ指令信号であるモータ指令値（目標速度）を入力し（ステップ S 2 5 7 1）、その目標速度に対応した整定時間（その対応関係は予め登録されている）を入力するとともに（ステップ S 2 5 7 2）、既に算出されている今回のシミ



ュレーション時間 $T_S (= \Delta t)$ を入力し（ステップ S 2 5 7 3）、今回のシミュレーション時間 $T_S (= \Delta t)$ での AC モータの回転角を計算する（ステップ S 2 5 7 4）。

#### 【 0 2 7 8 】

このとき、回転角は、目標速度×シミュレーション時間として算出できるが、モータ指令値が入力されてからのトータルのシミュレーション時間が整定時間以下であれば、回転角は、加減速を考慮して計算される。この後、その AC モータに連結した全ての関節（リンク）の関節値を計算して処理を終了する（ステップ S 2 5 7 5）。

#### 【 0 2 7 9 】

次に、マイクロスイッチをシミュレータ 3 0 でシミュレートする手法について、図 8 0 を参照しながら説明する。なお、図 8 0 は、マイクロスイッチのシミュレーション手順を説明するためのフローチャートである。

この図 8 0 に示すように、シミュレータ 3 0 によりマイクロスイッチのシミュレーションを行なう際には、まず、マイクロスイッチをセンサとして設定した関節の関節値を入力するとともに（ステップ S 2 5 8 1）、その関節値とセンサ値との対応関係（テーブル）を入力してから（ステップ S 2 5 8 2）、関節値とテーブルとに基づいてセンサ値（オン／オフ）を判定し（ステップ S 2 5 8 3）、そのセンサ値を組み込みソフトウェア開発部 4 0 に出力する（ステップ S 2 5 8 4）。

#### 【 0 2 8 0 】

次に、エンコーダをシミュレータ 3 0 でシミュレートする手法について、図 8 1 を参照しながら説明する。なお、図 8 1 は、エンコーダのシミュレーション手順を説明するためのフローチャートである。

この図 8 1 に示すように、シミュレータ 3 0 によりエンコーダのシミュレーションを行なう際には、まず、エンコーダをセンサとして設定した関節の関節値を入力するとともに（ステップ S 2 5 9 1）、エンコーダの分解能やエンコーダ原点での関節値を入力してから（ステップ S 2 5 9 2）、関節値に対応した波形を計算することにより関節値に対応したセンサ値を算出し（ステップ S 2 5 9 3）

、算出されたセンサ値を組み込みソフトウェア開発部 4 0 に出力する（ステップ S 2 5 9 4）。

【0 2 8 1】

次に、ポテンシオメータをシミュレータ 3 0 でシミュレートする手法について、図 8 2 を参照しながら説明する。なお、図 8 2 は、ポテンシオメータのシミュレーション手順を説明するためのフローチャートである。

この図 8 2 に示すように、シミュレータ 3 0 によりポテンシオメータのシミュレーションを行なう際には、まず、ポテンシオメータをセンサとして設定した関節の関節値を入力するとともに（ステップ S 2 6 0 1）、その関節値とセンサ値との対応関係（テーブル）を入力してから（ステップ S 2 6 0 2）、関節値とテーブルとに基づいてセンサ値を計算し（ステップ S 2 6 0 3）、そのセンサ値を組み込みソフトウェア開発部 4 0 に出力する（ステップ S 2 6 0 4）。

【0 2 8 2】

このように、本発明の一実施形態としての支援システム 1 0 によれば、3 次元リアルタイムシミュレーション装置 3 0 を中核に据えて、メカ試作品を作らなくても組み込みソフトウェア（制御プログラム）の開発をメカ設計とは単独に進めることができる。つまり、メカ設計部 2 0 による機構の設計と、その機構を制御するための組み込みソフトウェアの開発（組み込みソフトウェア開発部 4 0 による開発）とをコンカレントに遂行することができ、その組み込みソフトウェアの開発が効率化され、開発期間の短縮、工数の削減を実現することができる。

【0 2 8 3】

このとき、状態遷移図・表によるタスク制御記述法と 3 次元リアルタイムシミュレーション装置 3 0 とを組み合わせることにより、組み込みソフトウェアを試作品なしに効率よく開発することができる。

また、メカ設計部 2 0 と 3 次元リアルタイムシミュレーション装置 3 0 とを連携させることにより、製品を試作することなくシミュレーションによってその 3 次元機構モデルの可動部を動かし、部品どうしが不用意に接触せず所期の動作を行なうことができるかどうか等、その機構の確認を行ないながら、メカ設計を行なうことができる。その結果として、設計の不具合を早期に発見し、部品の配置

変更や可動部を考慮した修正等を行なうことが可能になり、製品開発に要する時間の短縮化やコスト削減を実現することができる。

【 0 2 8 4 】

さらに、マイコンチップ 4 2 と 3 次元リアルタイムシミュレーション装置 3 0 とを連携させることにより、機構を物理的に製作することなく、機構制御用のプログラムのデバックを行なうことができ、その組込みソフトウェアの開発の短縮化やコスト低減化に大きく寄与することになる。

なお、本発明は上述した実施形態に限定されるものではなく、本発明の趣旨を逸脱しない範囲で種々変形して実施することができる。

【 0 2 8 5 】

【発明の効果】

以上詳述したように、本発明の支援システム（請求項 1 ～ 1 0）および支援プログラムを記録したコンピュータ読み取り可能な記録媒体（請求項 1 1 ～ 1 8）によれば、3 次元機構モデルシミュレーション部を中核に据えて、メカ試作品を作らなくても組込みソフトウェアの開発をメカ設計とは単独に進めることができる。つまり、機構の設計と、その機構を制御するための組込みソフトウェアの開発とをコンカレントに遂行することができ、その組込みソフトウェアの開発が効率化され、開発期間の短縮、工数の削減を実現することができる。

【図面の簡単な説明】

【図 1】

本発明の一実施形態としての支援システムの機能的構成を示すブロック図である。

【図 2】

本実施形態のコンピュータ読み取り可能な記録媒体に記録された支援プログラムの構成を示す図である。

【図 3】

本発明の一実施形態としての支援システムを構築しうるコンピュータシステムを示す外観図である。

【図 4】

本発明の一実施形態としての支援システムを構築しうるコンピュータシステムのハードウェア構成を示すブロック図である。

【図 5】

本実施形態の支援システムの要部を示すブロック図である。

【図 6】

(A) および (B) は、それぞれ、メカ設計部から 3 次元リアルタイムシミュレーション装置へ受け渡される図形データファイルおよびアセンブリデータファイルの具体例を示す図である。

【図 7】

本実施形態の支援システムの要部（3 次元リアルタイムシミュレーション装置、状態遷移図・表作成編集装置および第 2 インタフェース部）を示すブロック図である。

【図 8】

状態遷移図の表示例を示す図である。

【図 9】

状態遷移表の表示例を示す図である。

【図 1 0】

(A) ～ (C) はマルチタスクについて説明するためのタイムチャートである。

【図 1 1】

(A) および (B) は優先度によるタスクのスケジューリングについて説明するための図である。

【図 1 2】

(A) ～ (D) は同期タスクを用いた同期化処理について説明するためのタイムチャートである。

【図 1 3】

本実施形態における同期タスクを用いた同期化処理について説明するためのブロック図である。

【図 1 4】

本実施形態における同期化処理について説明するためのフローチャートである。

【図 1 5】

本実施形態のロジックアナライザにおける表示例を示す図である。

【図 1 6】

ロジックアナライザによる処理を含む、本実施形態でのシミュレーション処理全体の流れを説明するためのフローチャートである。

【図 1 7】

本実施形態の支援システムにおける 3 次元リアルタイムシミュレーション装置の動作概念を説明するための概念図である。

【図 1 8】

センサやモータの定義時の処理を説明するための、3 次元リアルタイムシミュレーション装置の模式図である。

【図 1 9】

本実施形態の 3 次元リアルタイムシミュレーション装置によるシミュレーション処理を説明するための模式図である。

【図 2 0】

本実施形態の 3 次元リアルタイムシミュレーション装置と組込みソフトウェア開発部（同期タスク）との動作タイミングを説明するための図である。

【図 2 1】

本実施形態の 3 次元リアルタイムシミュレーション装置によるシミュレーション手順を説明するためのフローチャートである。

【図 2 2】

図 2 1 に示すシミュレータの処理フローの一部分（図 2 1 に一点鎖線で囲った部分）に代えて採用することのできる部分フローを示すフローチャートである。

【図 2 3】

モータとして定義されるリンク（モータリンク）の選び方を説明すべくモータリンクの一例を模式的に示す斜視図である。

【図 2 4】

(A)，(B) はセンサとして定義されるリンク（センサリンク）の選び方を説明すべくセンサリンクの一例を模式的に示す斜視図である。

【図 2 5】

(A)，(B) はセンサとして定義されるリンク（センサリンク）の選び方を説明すべくセンサリンクの他例を模式的に示す斜視図である。

【図 2 6】

モータの種類を説明するための図である。

【図 2 7】

モータを一次遅れ系とみなしたときの、モータの回転速度の初期変化を示す図である。

【図 2 8】

本実施形態の 3 次元リアルタイムシミュレーション装置における、一次遅れを考慮したモータ駆動の処理を説明するためのフローチャートである。

【図 2 9】

干渉チェックを用いたセンサの定義方法を説明するための図である。

【図 3 0】

干渉チェックを用いたセンサの定義方法を説明するための図である。

【図 3 1】

(A) ～ (C) は干渉チェックを用いたセンサの定義方法を説明するための図である。

【図 3 2】

本実施形態において 3 次元リアルタイムシミュレーション装置と組込みソフトウェア開発部との間で同期をとるために送受信される信号を説明するためのタイミングチャートである。

【図 3 3】

本実施形態において、3 次元機構モデルの動作とソフトウェア開発部の動作との同期をとるための同期化手法を説明するためのフローチャートである。

【図 3 4】

本実施形態において、3 次元機構モデルの動作とソフトウェア開発部の動作と

の同期を実現するための、3次元リアルタイムシミュレーション装置側の処理を説明するためのフローチャートである。

【図 3 5】

(A) , (B) は本実施形態での干渉可能性の評価方法の説明図である。

【図 3 6】

(A) ~ (C) は部品間距離とシミュレーション間隔との関係の各種の例を示す図である。

【図 3 7】

(A) , (B) は部品のグループ分けを説明するための図である。

【図 3 8】

(A) ~ (D) は1つの3次元機構モデル内に複数のモータリンクが定義されている場合のグループ分けを説明するための図である。

【図 3 9】

グループ分けした部品の再グループ化を説明するための図である。

【図 4 0】

複数のリンクを、モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを説明するためのフローチャートである。

【図 4 1】

図 4 0 における検索ルーチンを説明するためのフローチャートである。

【図 4 2】

図 4 0 および図 4 1 に示すルーチンの動作説明用のリンク機構モデルを模式的に示す斜視図である。

【図 4 3】

図 4 2 に示すリンク機構モデルのデータ構造を説明するための図である。

【図 4 4】

(A) ~ (D) は、モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法を具体的に説明するための図である。

【図 4 5】

本実施形態における別の実施態様の 3 次元リアルタイムシミュレーション装置によるシミュレーション処理全体の処理の流れを説明するためのフローチャートである。

【図 4 6】

モータ処理ルーチンを説明するためのフローチャートである。

【図 4 7】

リンク姿勢移動サブルーチンを説明するためのフローチャートである。

【図 4 8】

関節不良動作処理ルーチンを説明するためのフローチャートである。

【図 4 9】

センサ処理ルーチンを説明するためのフローチャートである。

【図 5 0】

オン／オフセンサ処理ルーチンを説明するためのフローチャートである。

【図 5 1】

センサ性能劣化処理ルーチンを説明するためのフローチャートである。

【図 5 2】

チャタリングによる出力予定値変更処理ルーチンを説明するためのフローチャートである。

【図 5 3】

エンコーダの変位（角度等）に対する出力波形の一例を示す図である。

【図 5 4】

(A) ～ (E) は、エンコーダの、各種の出力波形を示す図である。

【図 5 5】

エンコーダの出力処理ルーチンを説明するためのフローチャートである。

【図 5 6】

エンコーダ処理ルーチンを説明するためのフローチャートである。

【図 5 7】

ポテンシオメータの関数値の一例を説明するための図である。



【図 5 8】

ポテンシオメータ前処理ルーチンを説明するためのフローチャートである。

【図 5 9】

シミュレーション間隔決定処理ルーチンを説明するためのフローチャートである。

【図 6 0】

(A) , (B) は干渉チェックによる関節可動範囲の検索法を説明するための図である。

【図 6 1】

(A) ~ (C) は干渉チェックによるカム関係の検索方法を説明するための図である。

【図 6 2】

干渉チェックによるカム関係検索ルーチンを説明するためのフローチャートである。

【図 6 3】

受動部品の非接触姿勢検索ルーチンを説明するためのフローチャートである。

【図 6 4】

受動部品の非接触姿勢検索ルーチンを説明するためのフローチャートである。

【図 6 5】

(A) ~ (C) は駆動部材側の溝とその溝に挿入されて駆動される受動部材側のピンとの溝関係を設定する手法を説明するための図である。

【図 6 6】

ギアやローラをリンクとして有する場合のリンク姿勢移動ルーチンを説明するためのフローチャートである。

【図 6 7】

関節値移動サブルーチンを説明するためのフローチャートである。

【図 6 8】

関節値初期化サブルーチンを説明するためのフローチャートである。

【図 6 9】

(A) ～ (C) はギアやローラの動作について説明するための図である。

【図 7 0】

ローラをリンクとして有する場合の関節移動サブルーチンの変更部分を説明するためのフローチャートである。

【図 7 1】

(A) ～ (F) は、機構モデルとしての昇降機とその昇降機を成すリンクのツリー構造とを説明するための図である。

【図 7 2】

各リンクに持たせる、親子関係の切替情報を示す図である。

【図 7 3】

親切替チェックサブルーチンを説明するためのフローチャートである。

【図 7 4】

DCモータのシミュレーション手順を説明するためのフローチャートである。

【図 7.5】

DCモータの入力値とモータ指令値との対応関係の一例を示す図である。

【図 7 6】

ステッピングモータのシミュレーション手順を説明するためのフローチャートである。

【図 7 7】

ステッピングモータの加速パターンを説明するための図である。

【図 7 8】

パルス数計算サブルーチンを説明するためのフローチャートである。

【図 7 9】

ACモータのシミュレーション手順を説明するためのフローチャートである。

【図 8 0】

マイクロスイッチのシミュレーション手順を説明するためのフローチャートである。

【図 8 1】

エンコーダのシミュレーション手順を説明するためのフローチャートである。

【図 8 2】

ポテンショメータのシミュレーション手順を説明するためのフローチャートである。

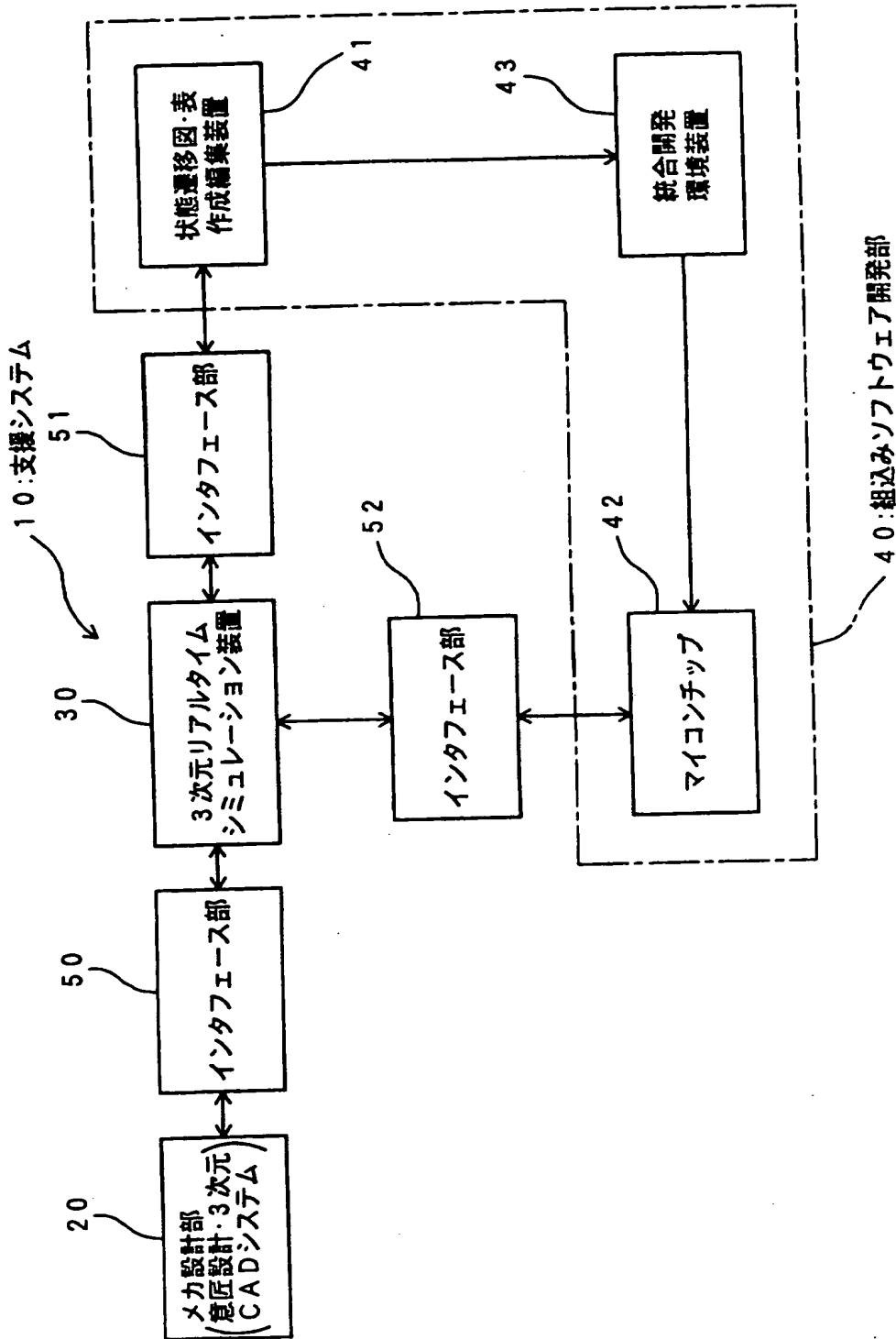
【符号の説明】

- 1 0 支援システム
- 2 0 メカ設計部（機構設計部，意匠設計・3次元CADシステム）
- 3 0 3次元リアルタイムシミュレーション装置（3次元機構モデルシミュレーション部，3Dモデルシミュレータ）
  - 3 1 シミュレーション部
  - 3 2 インタフェース部
- 4 0 組込みソフトウェア開発部
  - 4 1 状態遷移図・表作成編集装置（状態遷移図・表作成部）
    - 4 1 1 タスク制御部
    - 4 1 2 ユーザインタフェース部
    - 4 1 3 状態遷移表示部
    - 4 1 4 インタフェース部
    - 4 1 5 内部メモリ
  - 4 2 マイコンチップ
  - 4 3 統合開発環境装置
- 5 0 第1インタフェース部
  - 5 1, 5 2 第2インタフェース部
  - 5 3 共有メモリ
- 1 0 0 コンピュータシステム
  - 1 0 1 本体部
    - 1 0 1 a フロッピーディスク装填口
    - 1 0 1 b CD-ROM装填口
  - 1 0 2 CRTディスプレイ
    - 1 0 2 a 表示画面
  - 1 0 3 キーボード

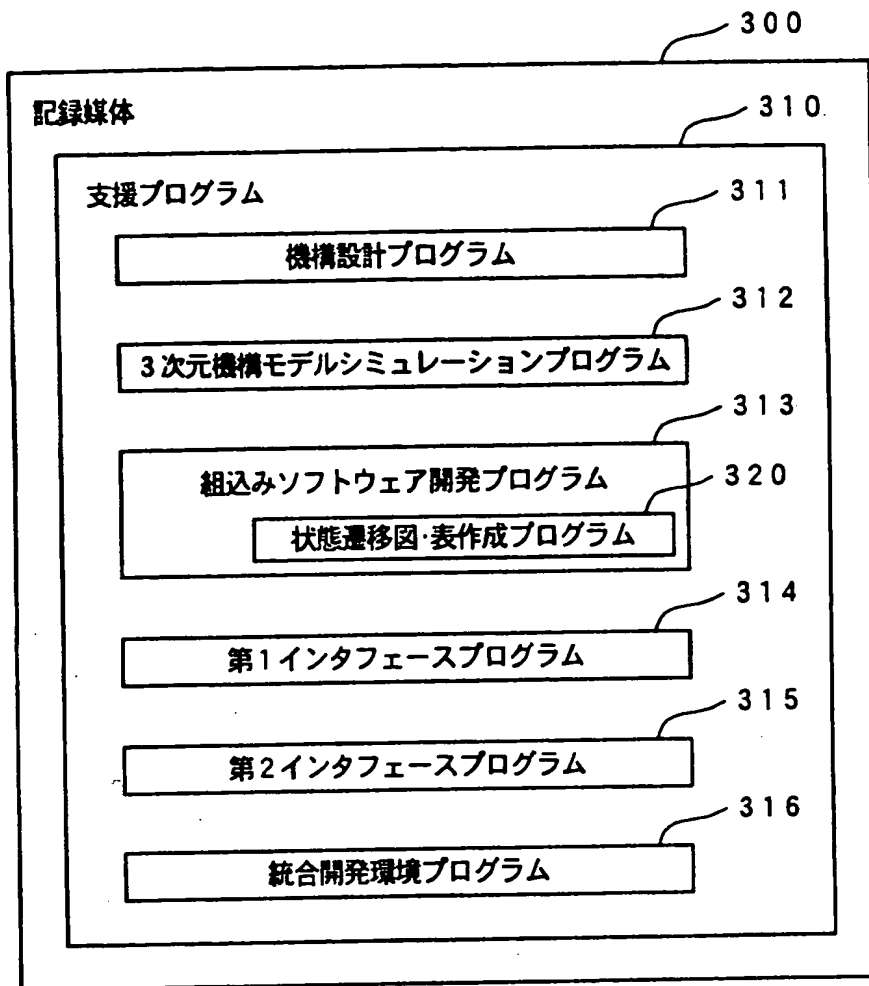
- 104 マウス
- 210 CD-ROM
- 211 ハードディスク
- 212 フロッピーディスク
- 220 バスライン
- 221 中央演算処理装置 (CPU)
- 222 RAM
- 223 ハードディスクコントローラ
- 224 フロッピーディスクドライバ
- 225 CD-ROMドライバ
- 226 マウスコントローラ
- 227 キーボードコントローラ
- 228 ディスプレイコントローラ
- 229 モデム
- 230 ROM
- 300 記録媒体
- 310 支援プログラム
- 311 機構設計プログラム
- 312 3次元機構モデルシミュレーションプログラム
- 313 組込みソフトウェア開発プログラム
- 314 第1インタフェースプログラム
- 315 第2インタフェースプログラム
- 316 統合開発環境プログラム
- 320 状態遷移図・表作成プログラム

【書類名】 図面

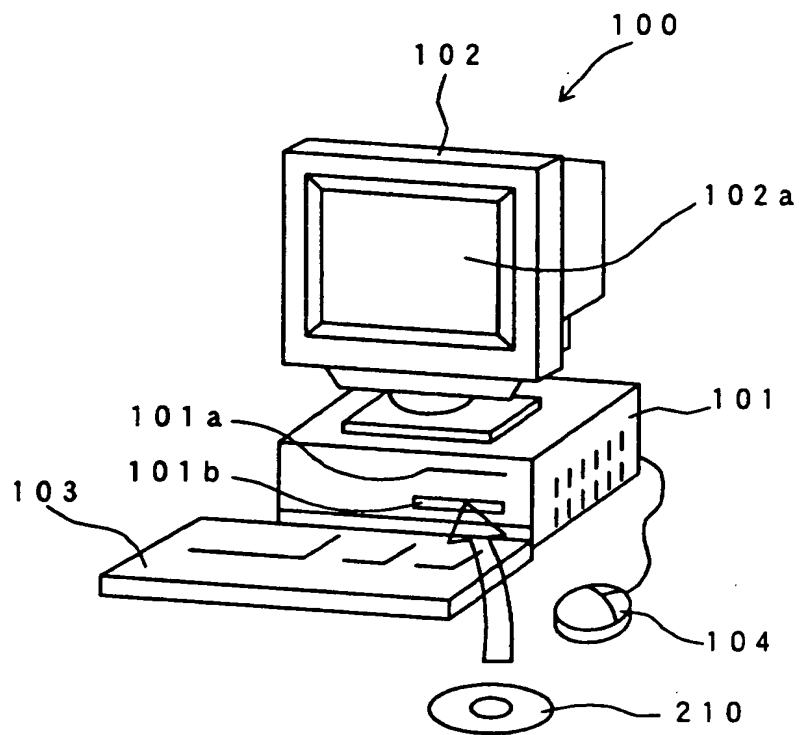
【図 1】



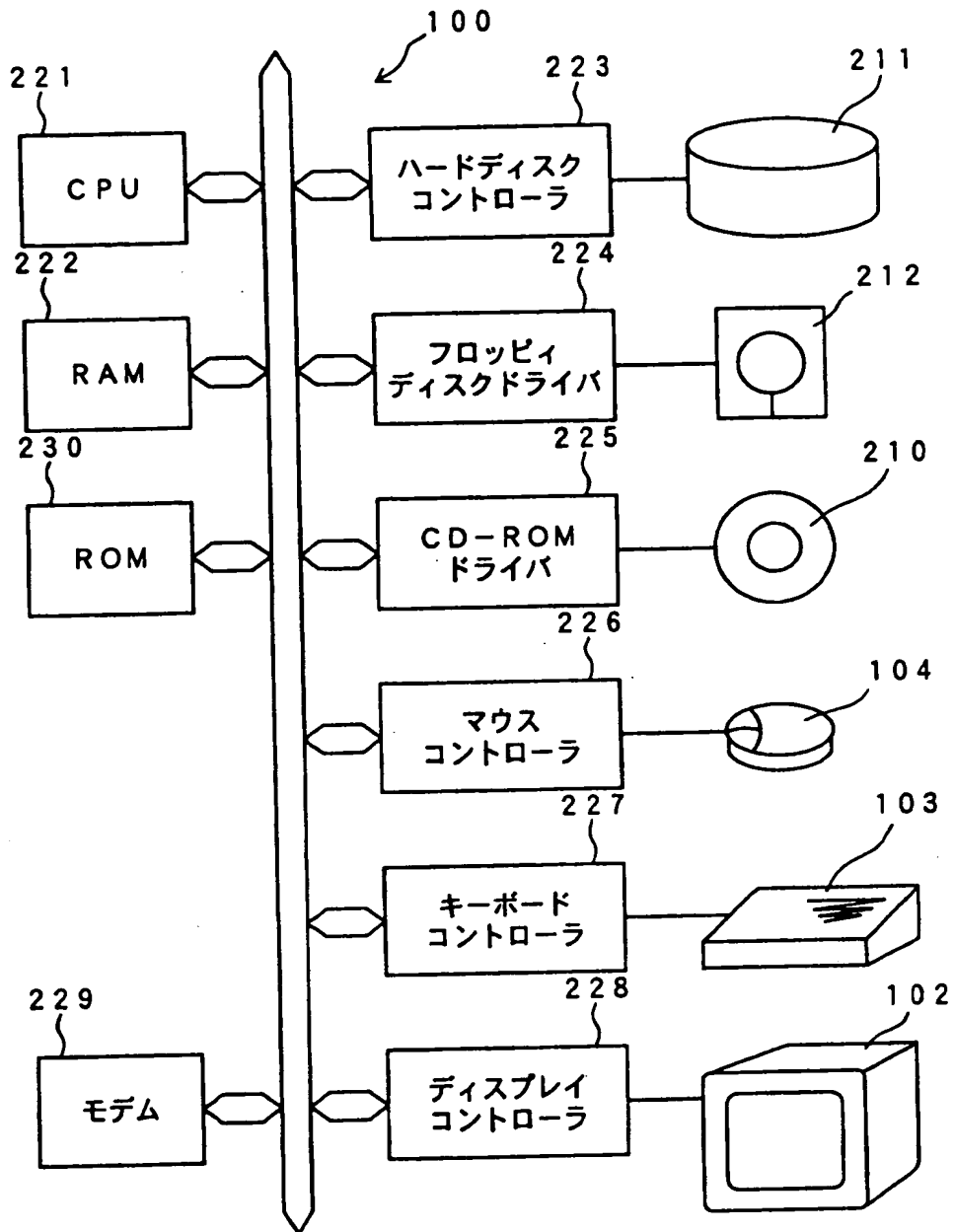
【図 2】



【図 3】

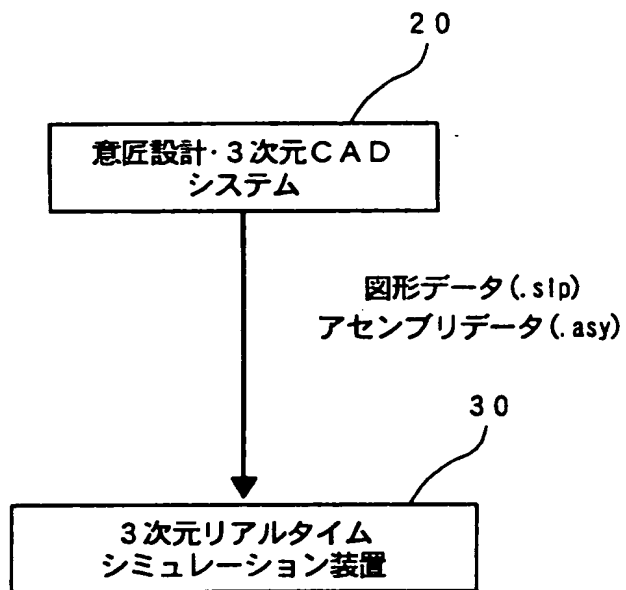


【図4】





【図 5】



【図 6】

(A)

```

solid Kam_Version2__0_Library1
  color 1.000000 0.660000 0.000000
  facet
    normal 0.000000 0.952236 -0.305364
    normal 0.000000 0.952236 -0.305364
    normal 0.000000 0.952236 -0.305364
  outer loop
    vertex 46.667969 -54.467480 -31.538586
    vertex 46.667969 -57.256718 -40.236450
    vertex 16.667969 -57.256718 -40.236450
  endloop
endfacet
facet
  normal 0.000000 0.952236 -0.305364
  normal 0.000000 0.952236 -0.305364
  normal 0.000000 0.952236 -0.305364
  outer loop
    vertex 16.667969 -57.256718 -40.236450
    vertex 16.667969 -54.467480 -31.538586
    vertex 46.667969 -54.467480 -31.538586
  endloop
endfacet
.
.
.
facet
  normal 0.000000 -0.472754 0.881194
  normal 0.000000 -0.472754 0.881194
  normal 0.000000 -0.472754 0.881194
  outerloop
    vertex 91.667969 -77.803757 -26.071329
    vertex 91.667969 -73.224937 -23.614828
    vertex 46.667969 -73.224937 -23.614828
  endloop
endfacet
endsolid Kam_Version2__0_Library1

```

(B)

```

Begin Space
# No 1
Parent 0 # root
JointType Free
End

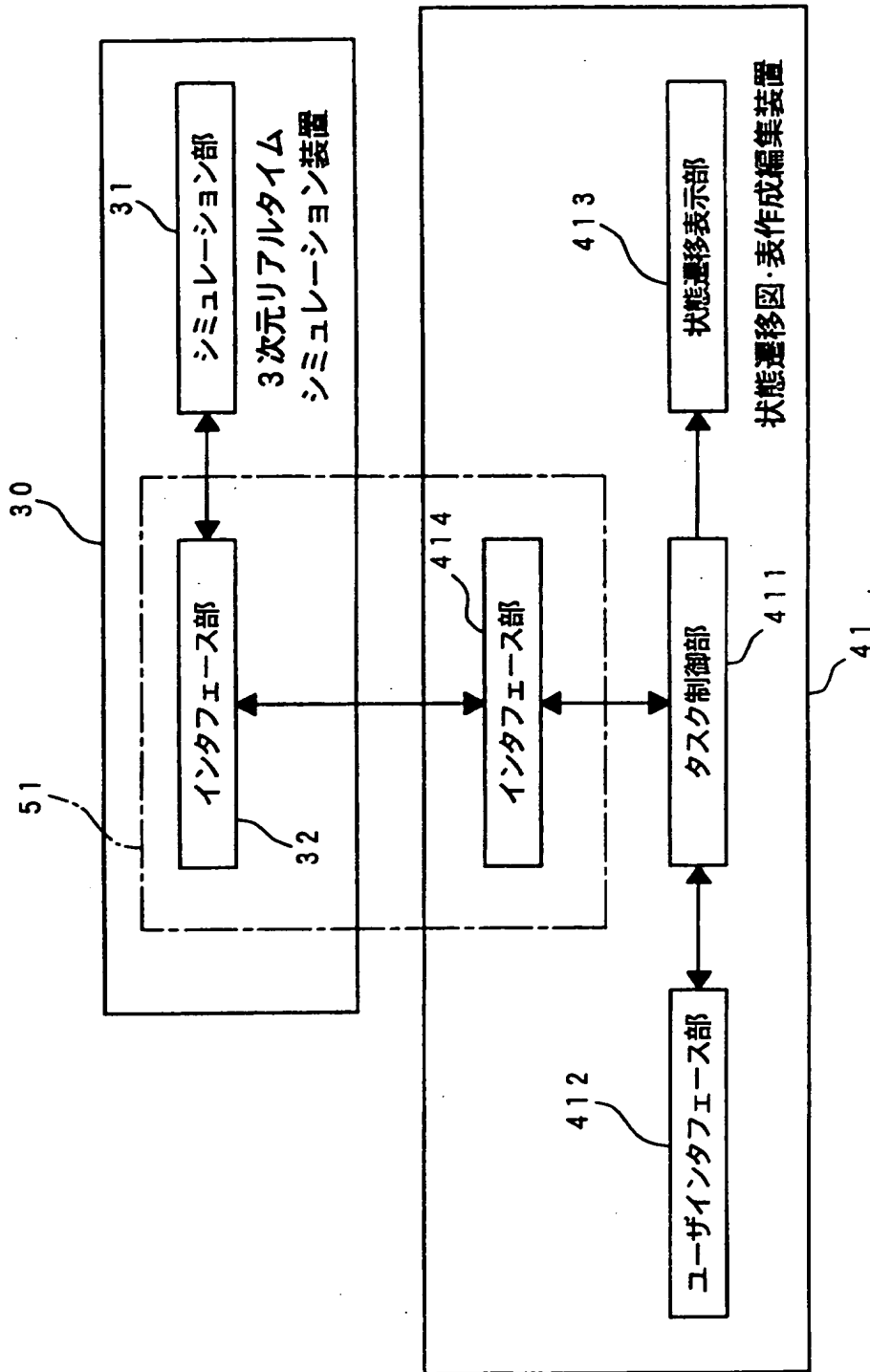
Begin "seat_assy__0_Library1"
# No 2
Parent 1 # Space
RelPosition 0.000000 0.000000 0.000000
RelAttitude 1.000000 0.000000 0.000000 0.000000
1.000000 0.000000 0.000000 0.000000 1.000000
JointType Fixed
End

Begin "seat_pan__0_Library1"
# No 3
Parent 2 # seat_assy__0_Library1
Shape seat_pan__0_Library1.slp
RelPosition 1.810953 51.404652 450.053009
RelAttitude 1.000000 0.000000 0.000000 0.000000
1.000000 0.000000 0.000000 0.000000 1.000000
JointType Fixed
End

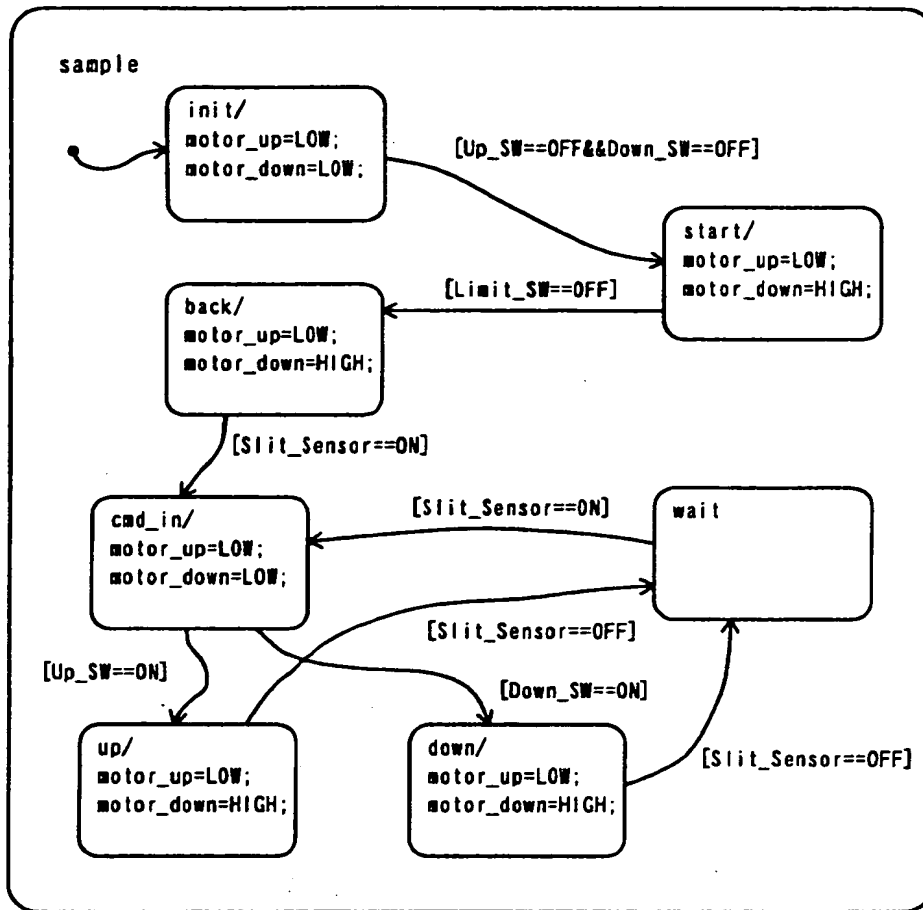
Begin "Kam_Version2__0_Library1"
# No 4
Parent 2 # seat_assy__0_Library1
Shape Kam_Version2__0_Library1.slp
RelPosition 201.712204 -211.475098 811.809509
RelAttitude 0.104385 0.034423 -0.993941 0.994522
-0.009110 0.104131 -0.005471 -0.999366 -0.035185
JointType Fixed
End
.
.
.

```

【図 7】



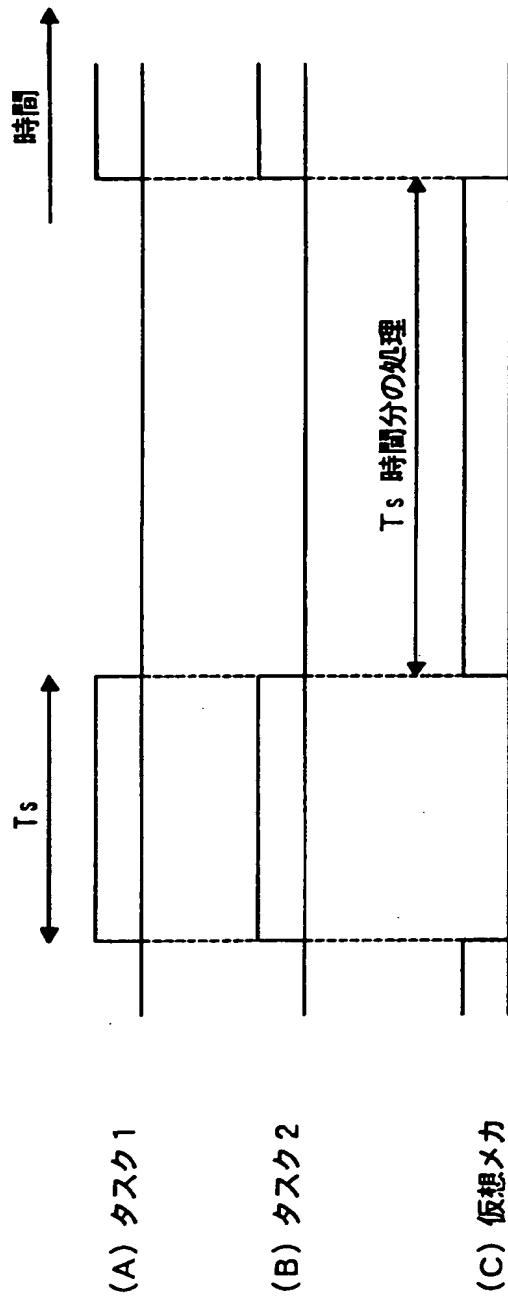
【図 8】



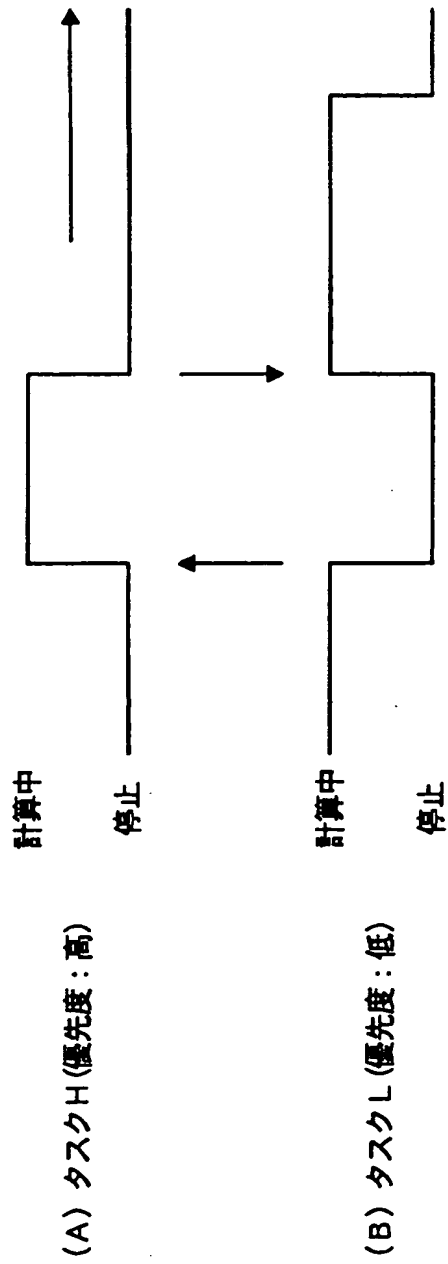
【図 9】

		停止状態	直進中	左カーブ中	右カーブ中
	$\begin{matrix} S \\ E \end{matrix}$	0	1	2	3
信号__青	0	$\frac{1}{\text{アセムON()};}$	$\frac{-}{/}$	$\frac{-}{/}$	$\frac{-}{/}$
左カーブ	1	$\frac{-}{/}$	$\frac{2}{\text{アセムOFF()};}$ 左旋回();	$\frac{-}{/}$ 左旋回();	$\frac{2}{/}$ 左旋回();
右カーブ	2	$\frac{-}{/}$	$\frac{3}{\text{アセムOFF()};}$ 右旋回();	$\frac{3}{/}$ 右旋回();	$\frac{3}{/}$ 右旋回();
直線	3	$\frac{-}{/}$	$\frac{-}{/}$	$\frac{1}{\text{ハンドルを戻す()};}$ アセムON();	$\frac{1}{\text{ハンドルを戻す()};}$ アセムON();

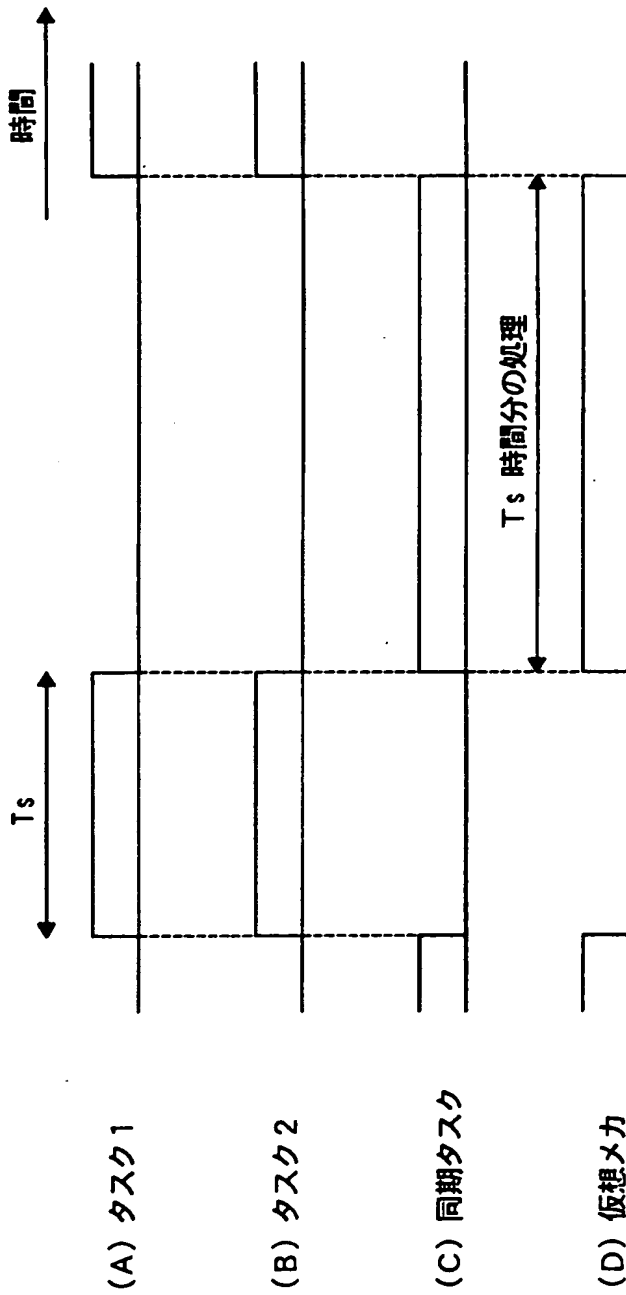
【図 1 0】



【図 1 1】

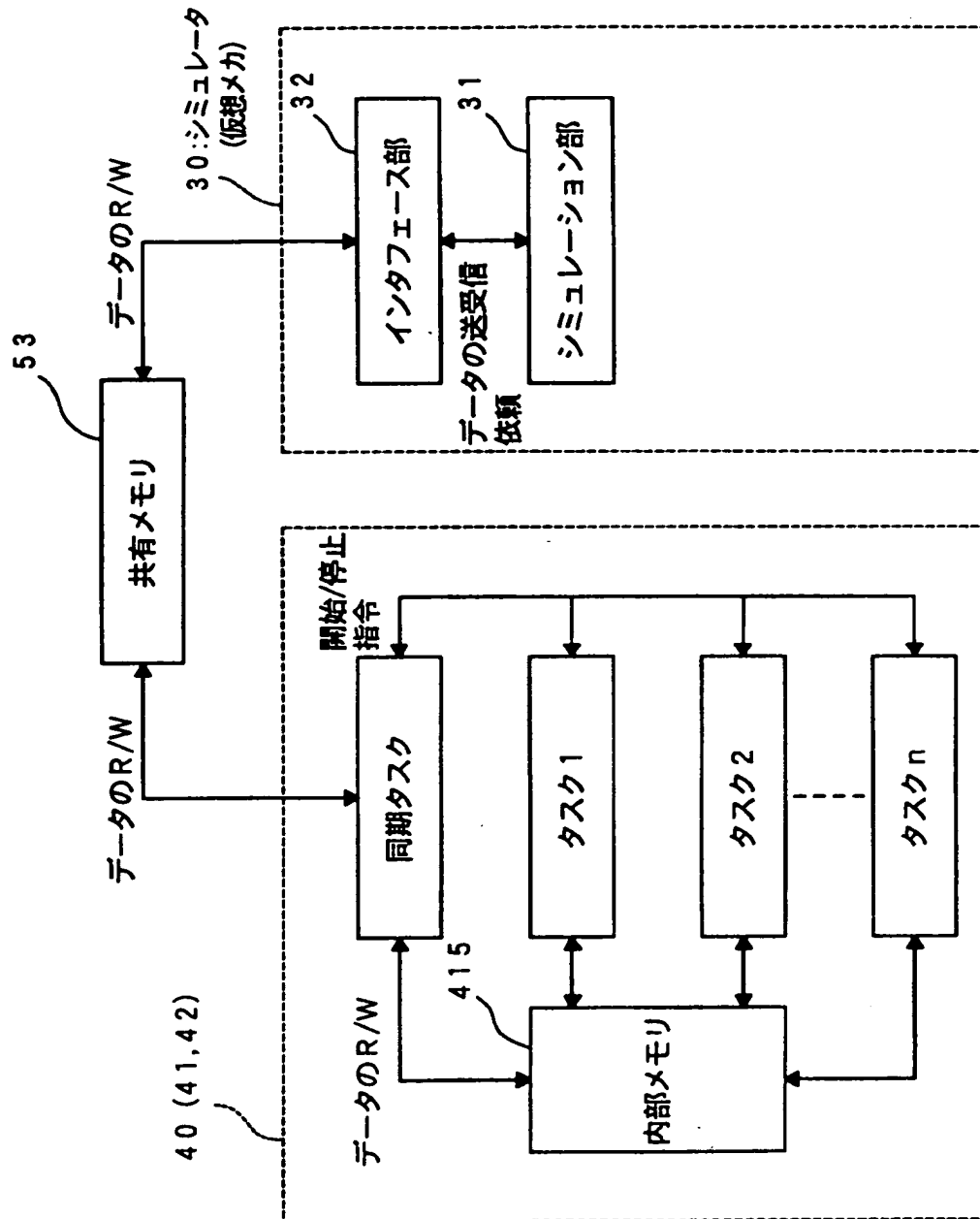


【図 1 2】

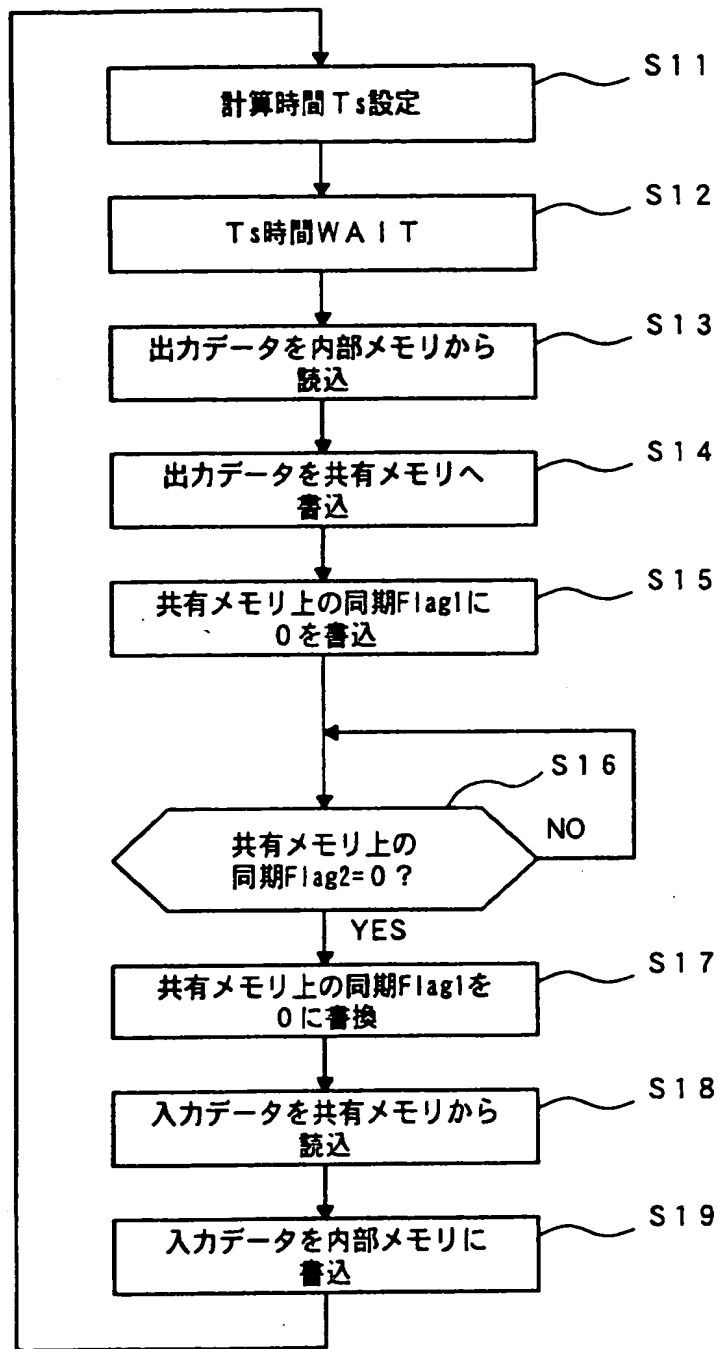




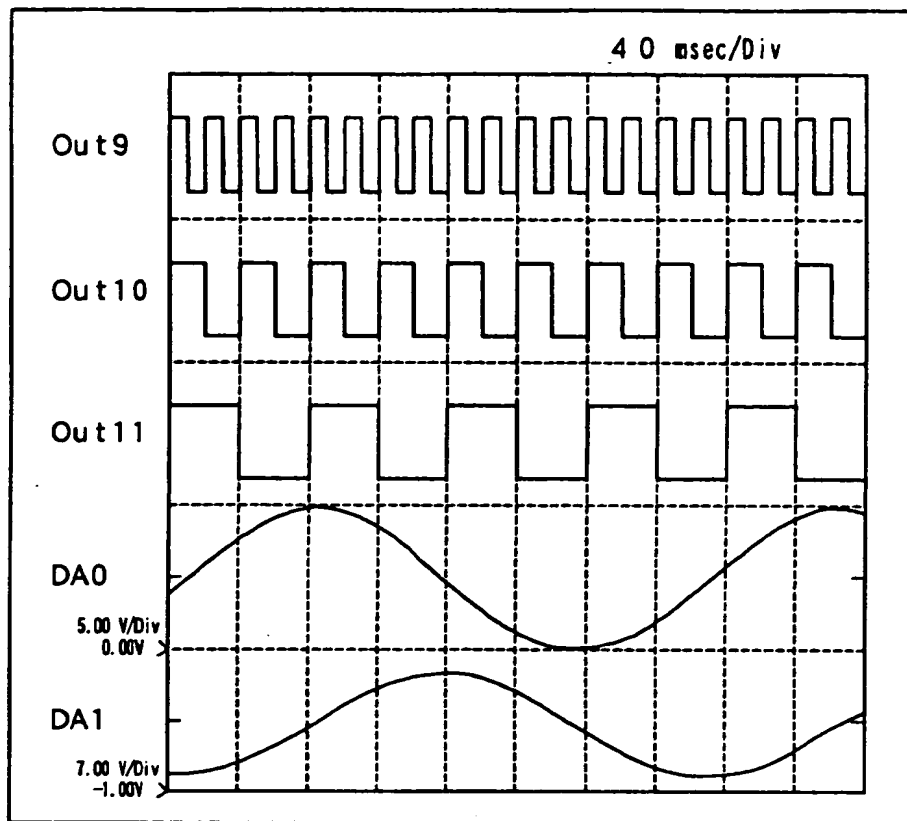
【図 13】



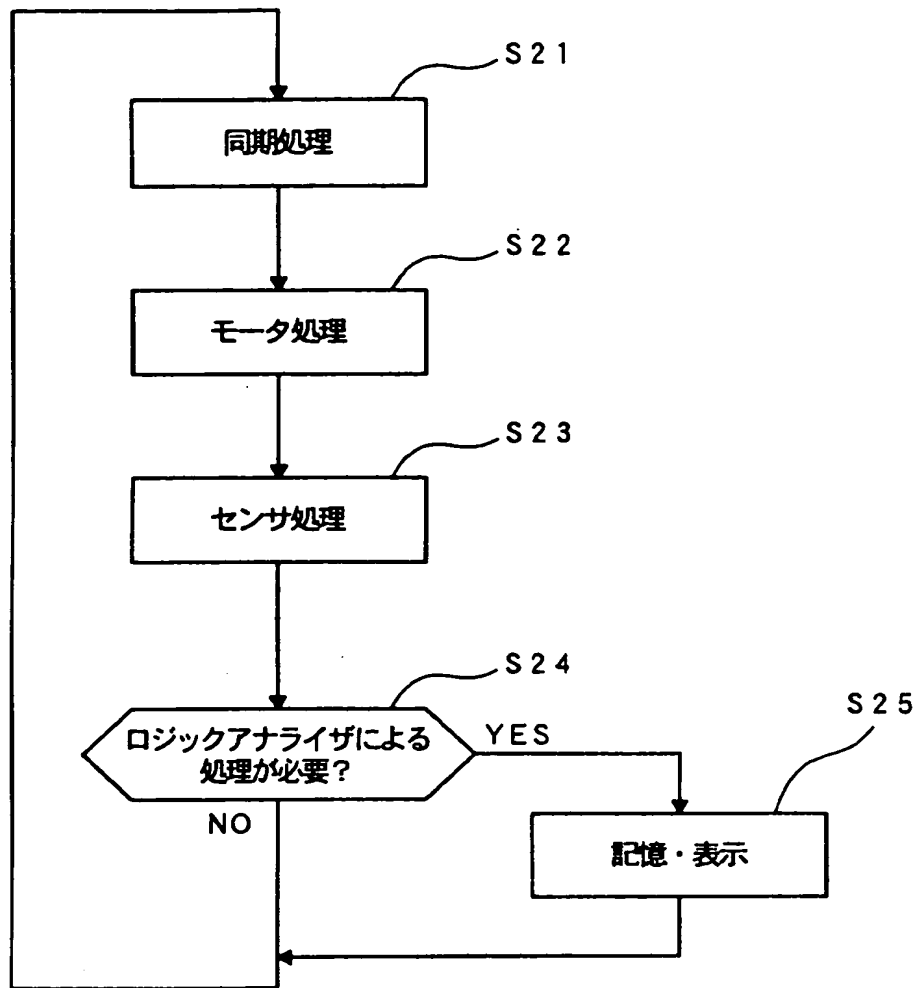
【図 1 4】



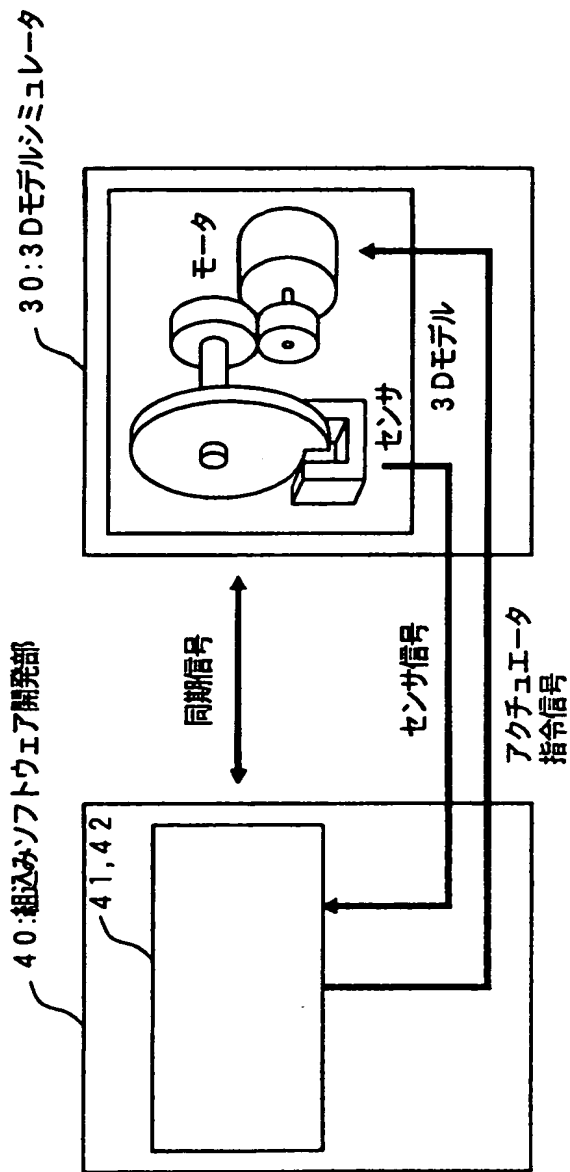
【図 15】



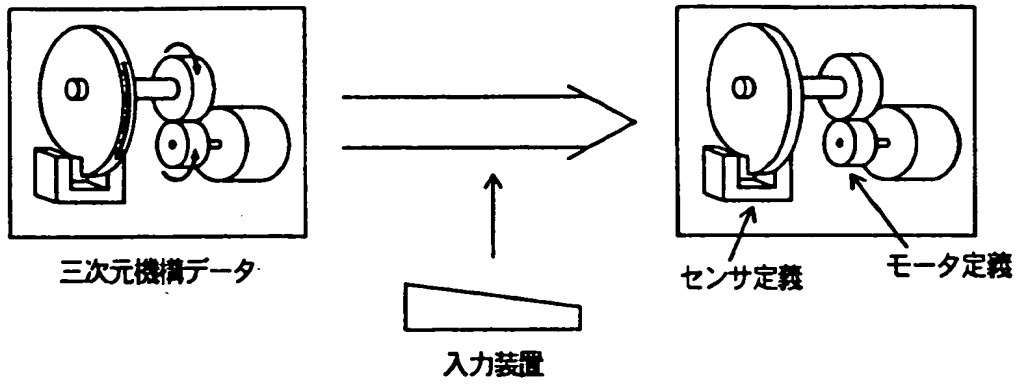
【図 1 6】



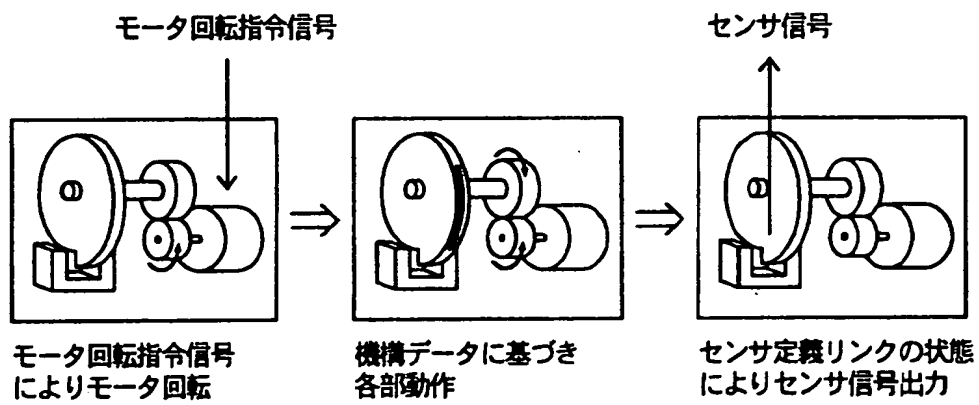
【図 17】



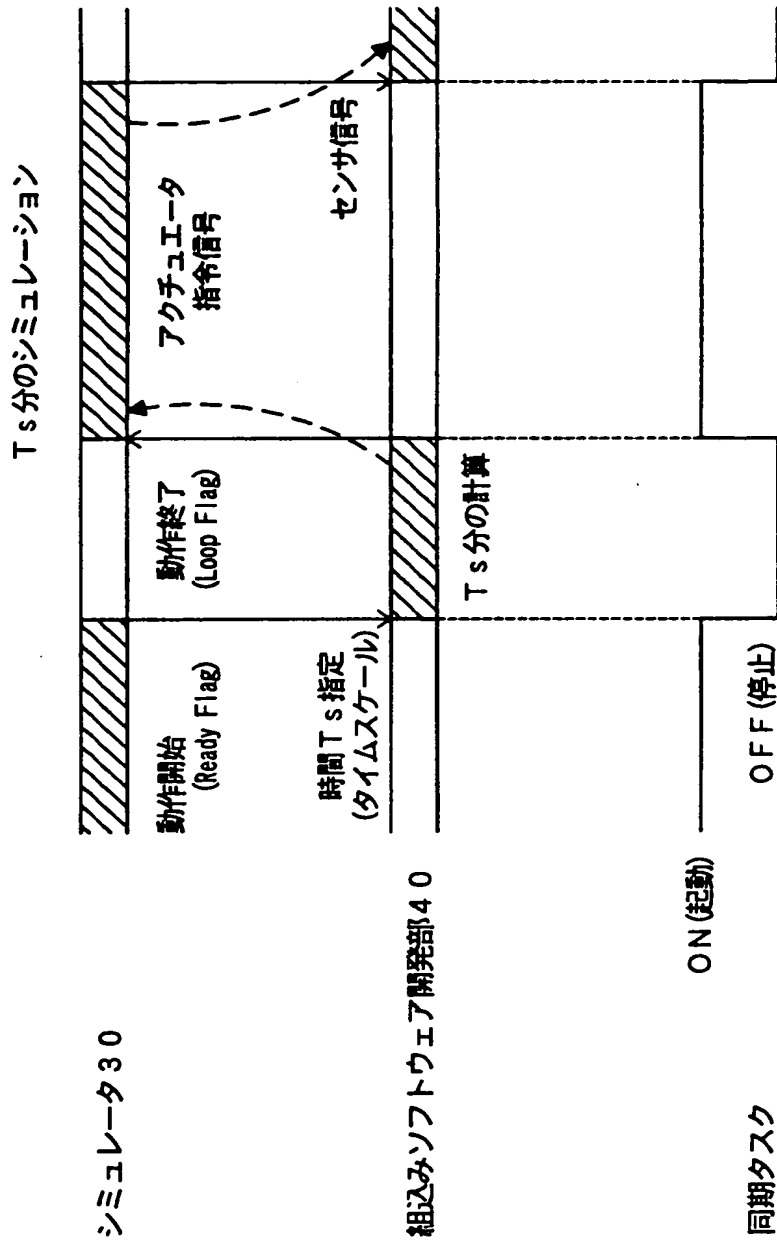
【図 18】



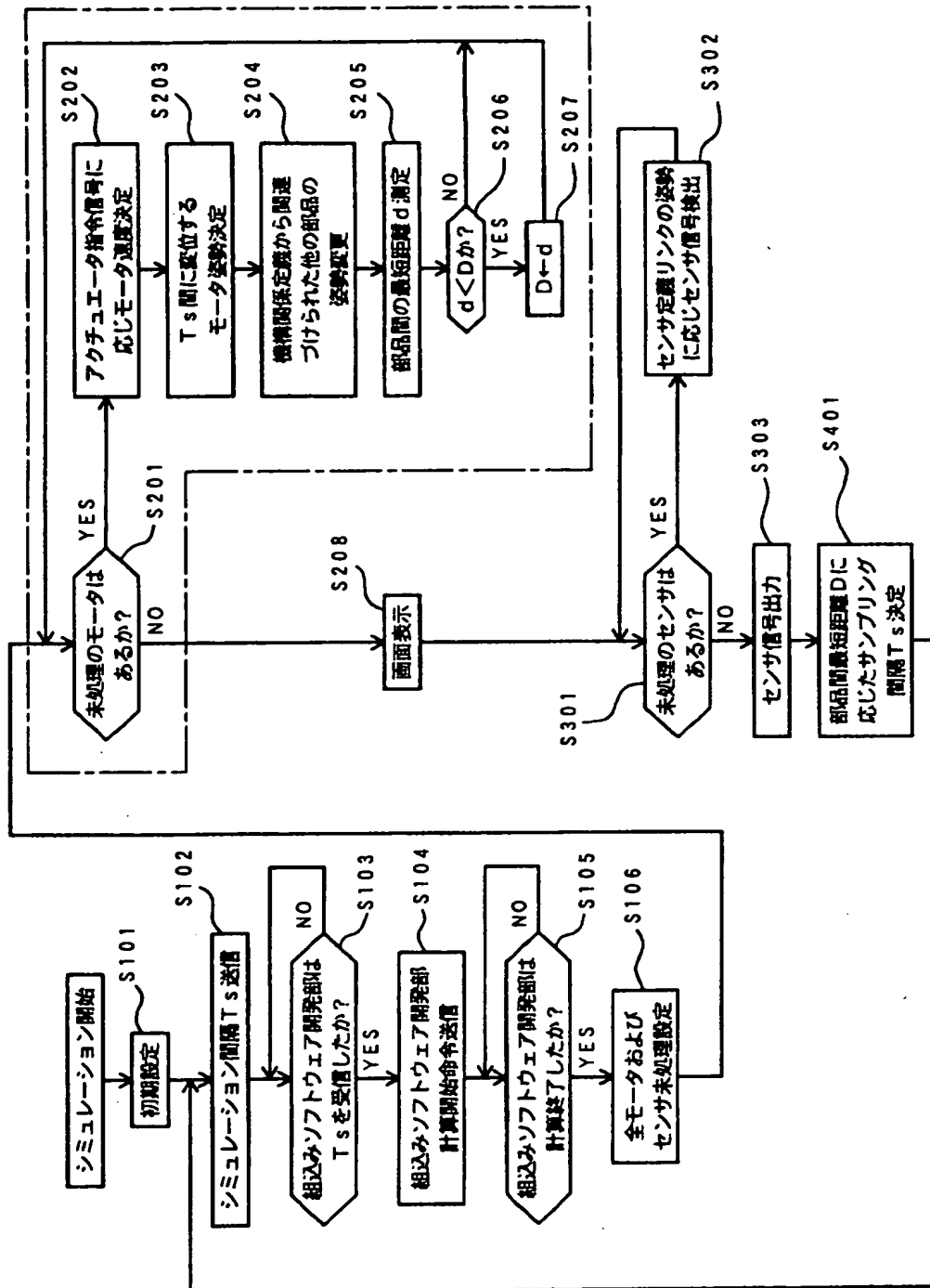
【図 19】



【図 2 0】

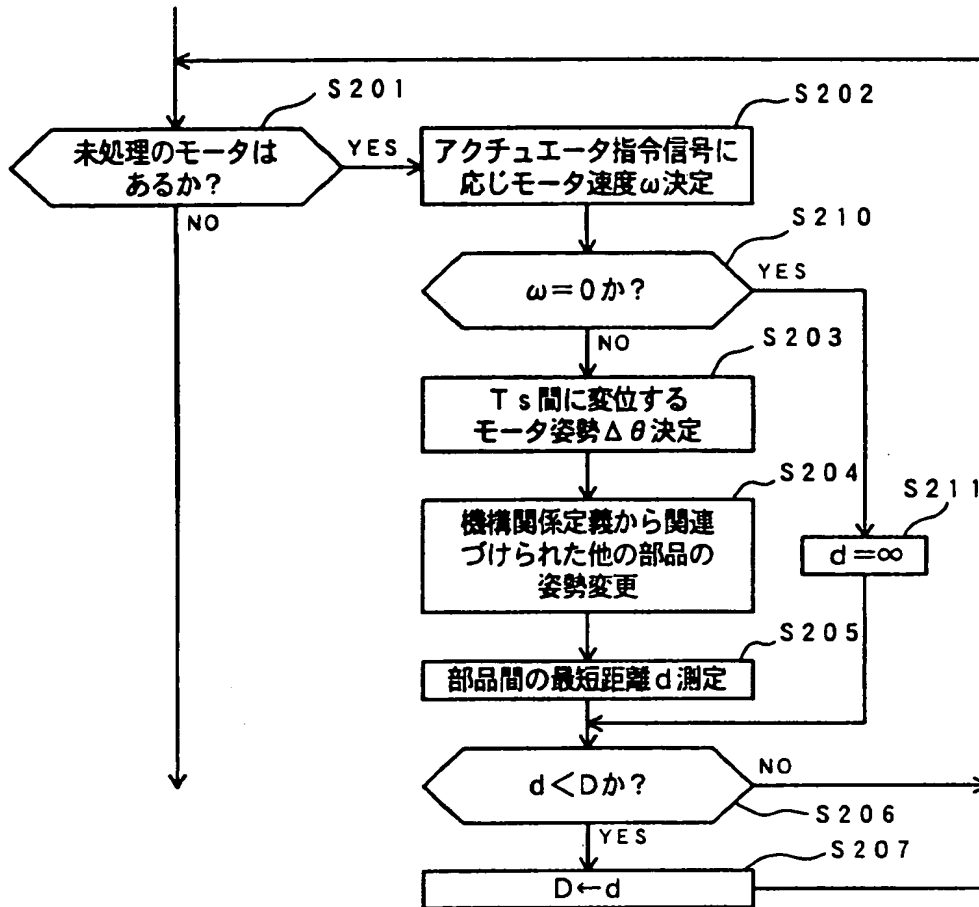


【図 21】

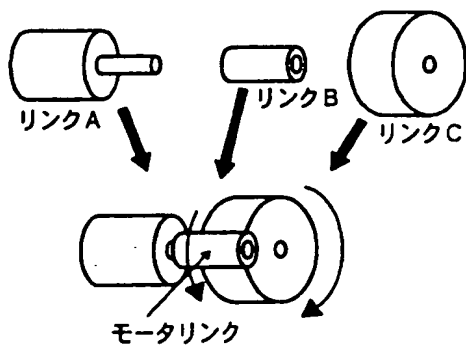




【図 22】

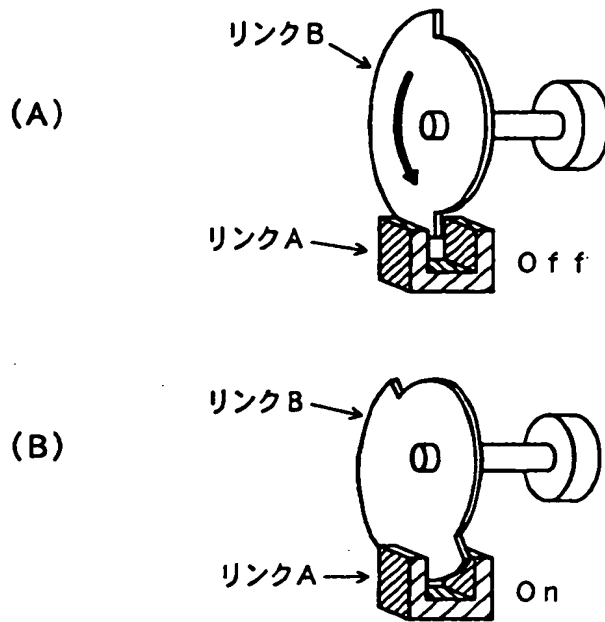


【図 23】

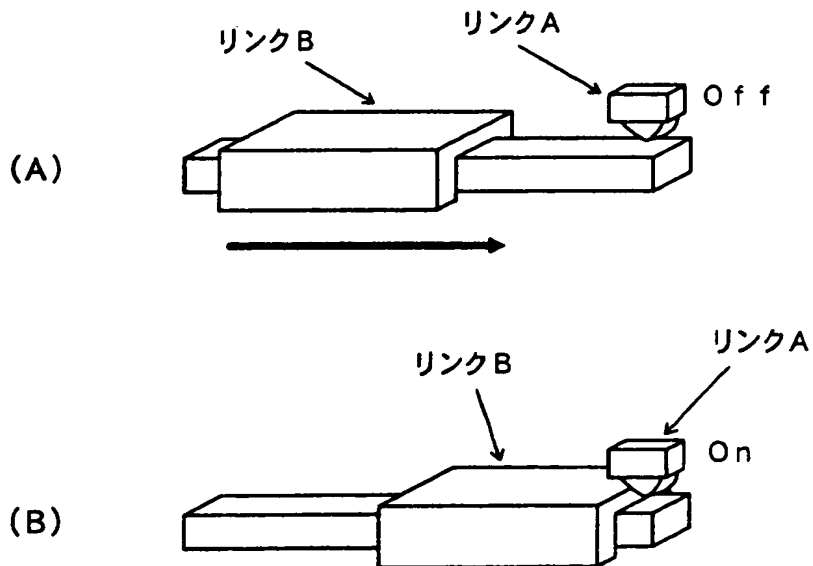




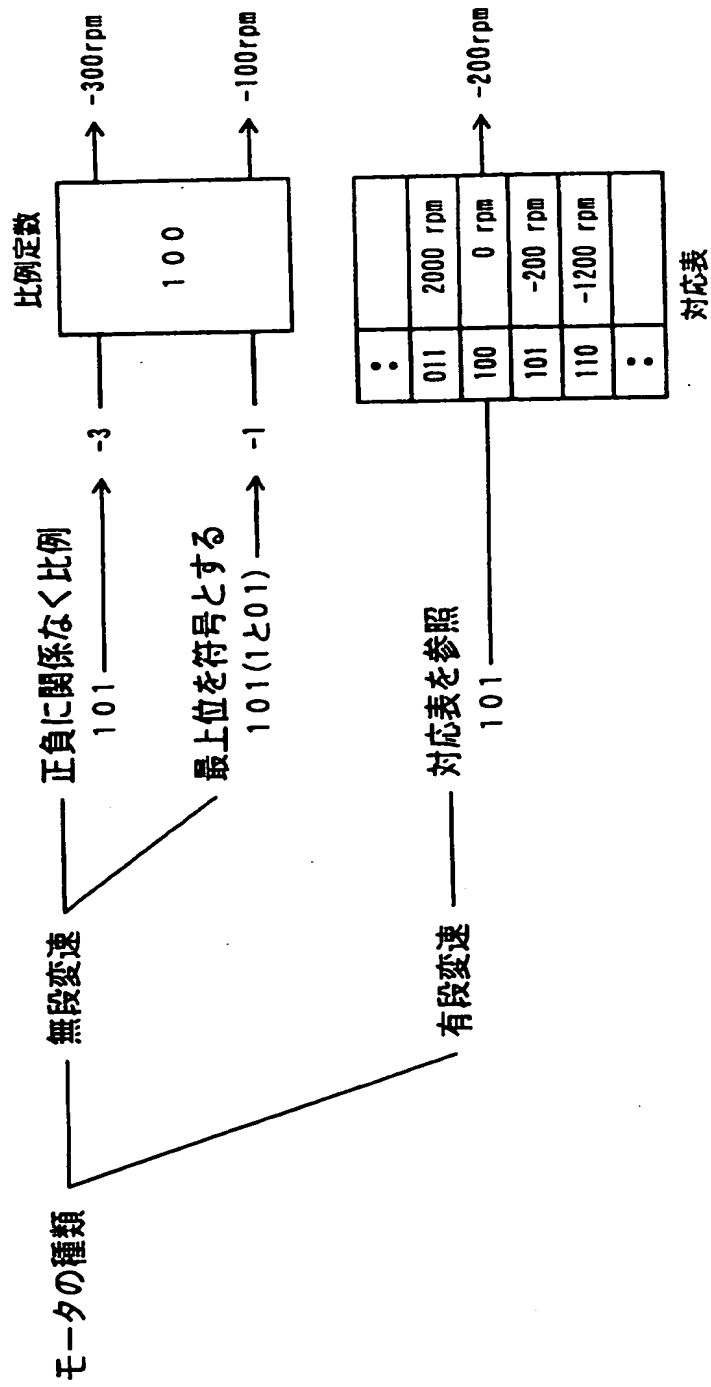
【図 2 4】



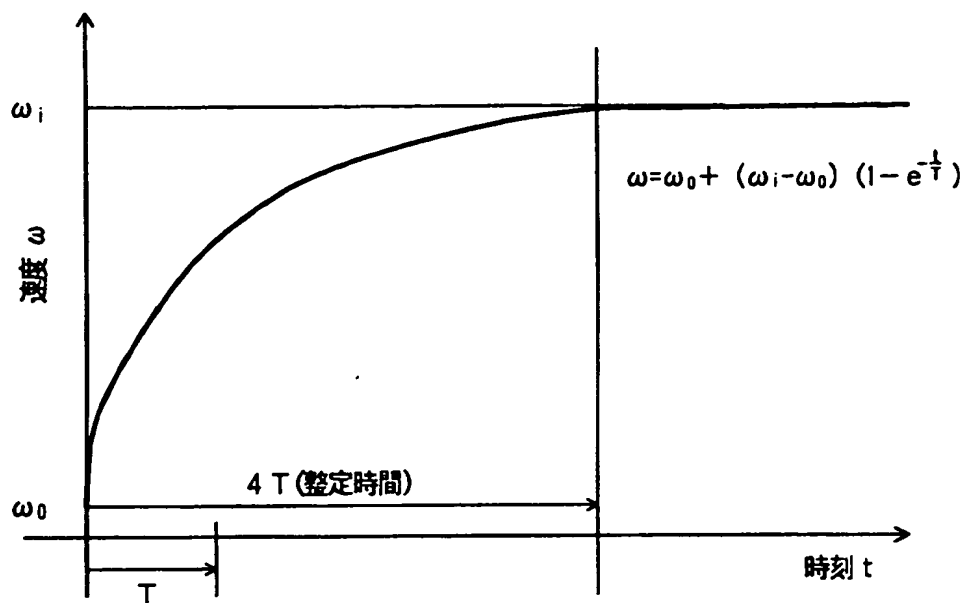
【図 2 5】



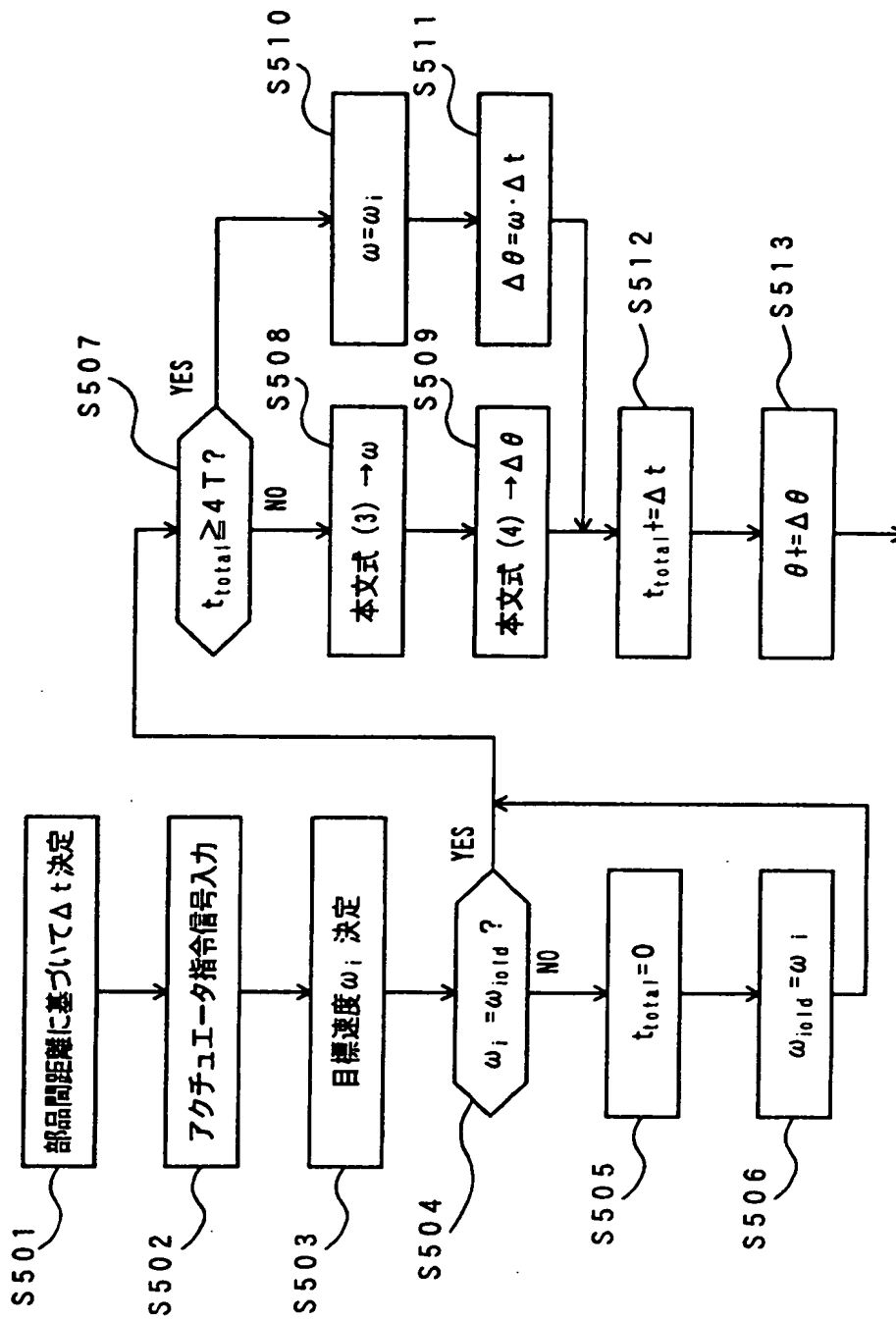
【図 2 6】



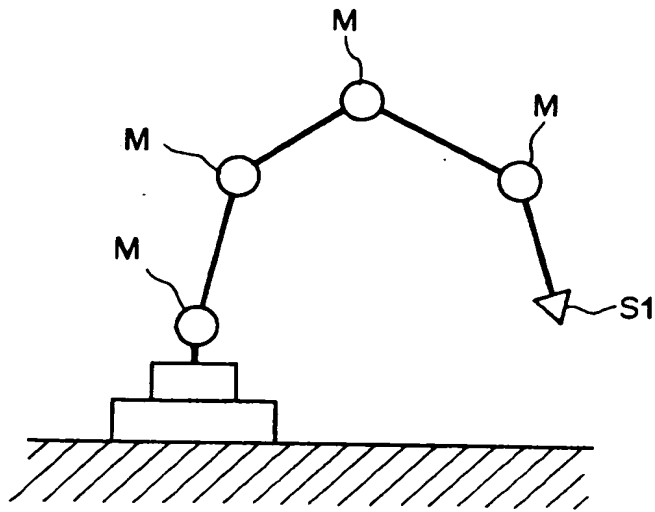
【図 2 7】



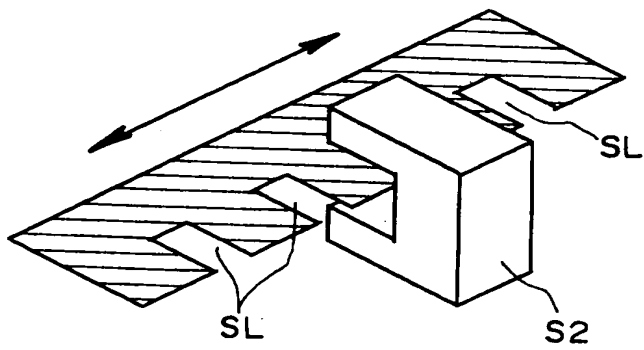
【図 28】



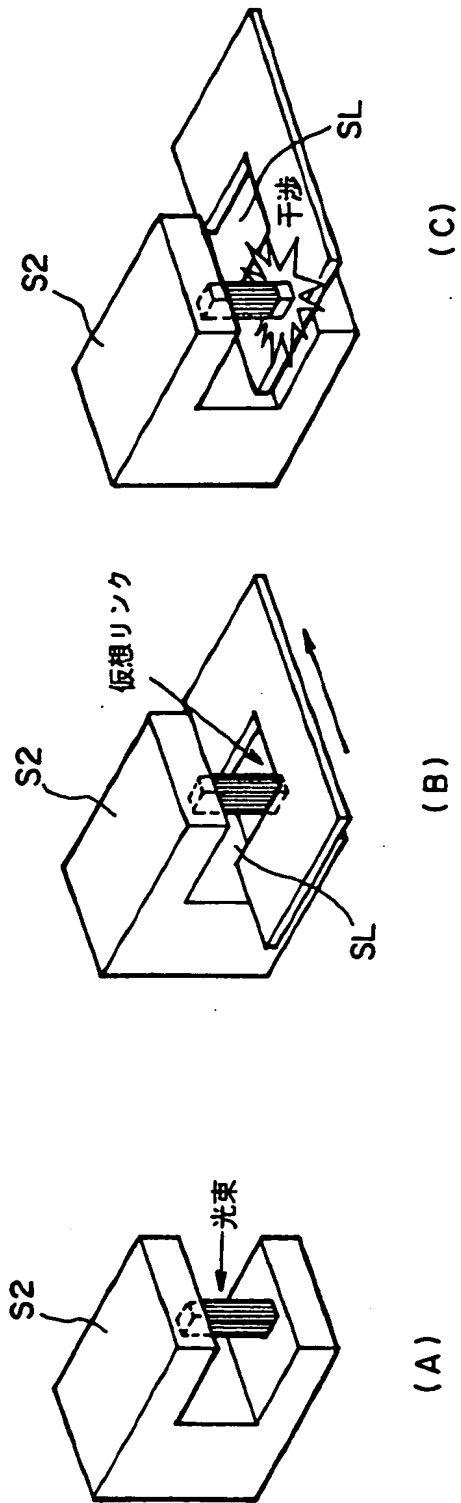
【図 29】



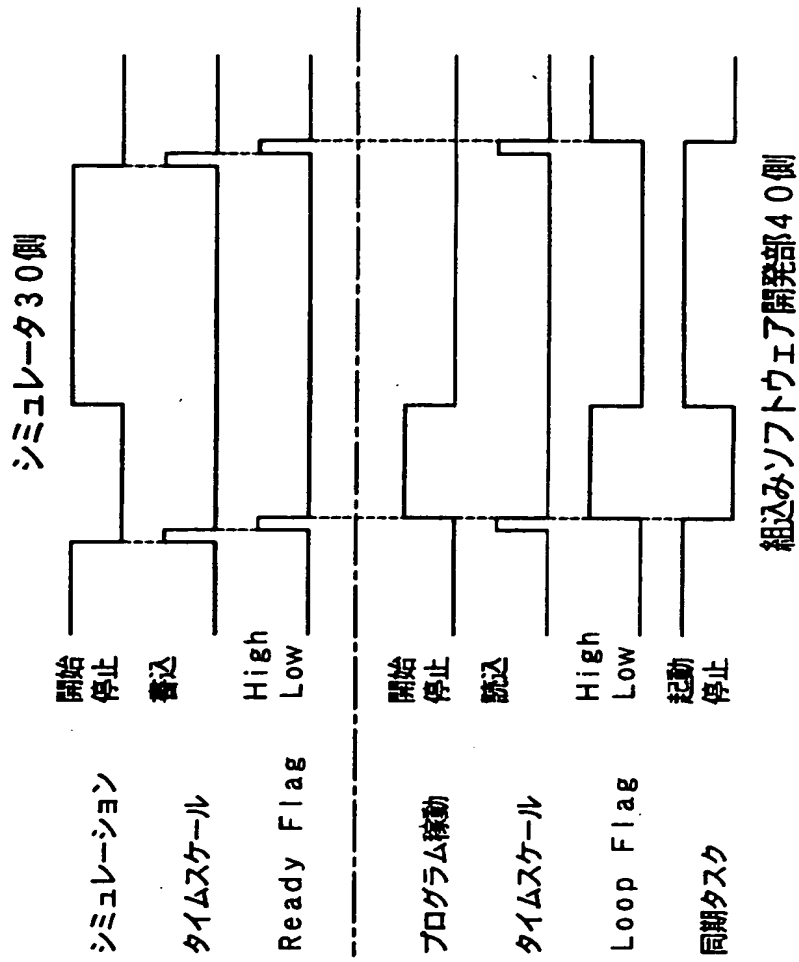
【図 30】



【図 31】

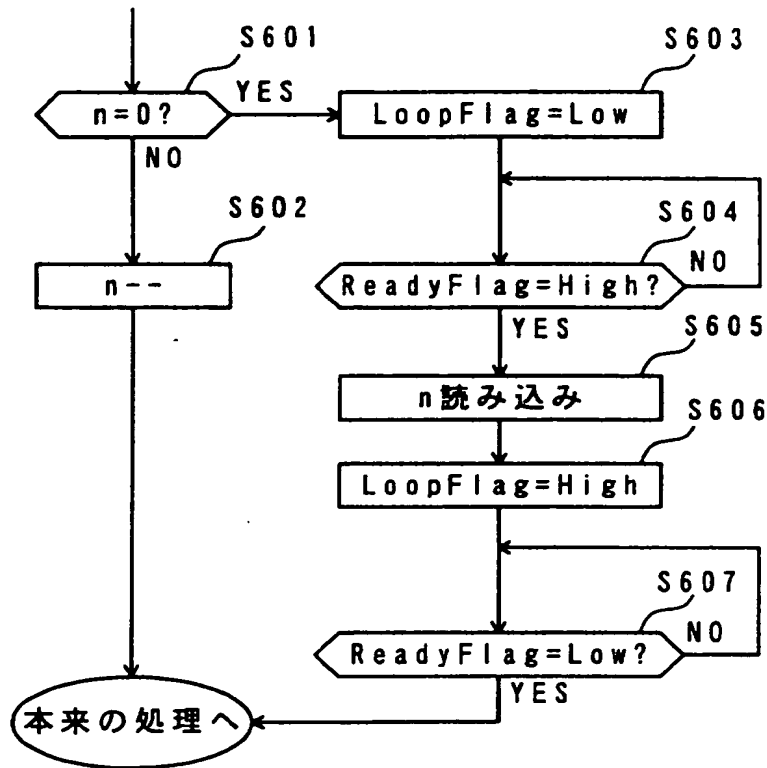


【図 3 2】

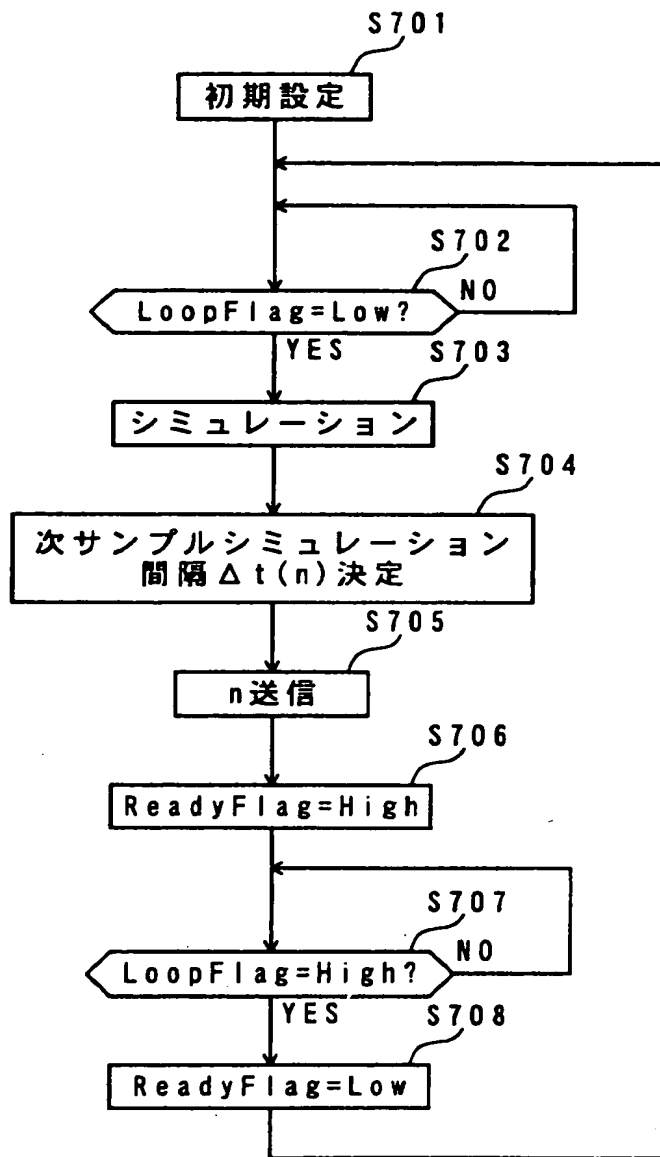




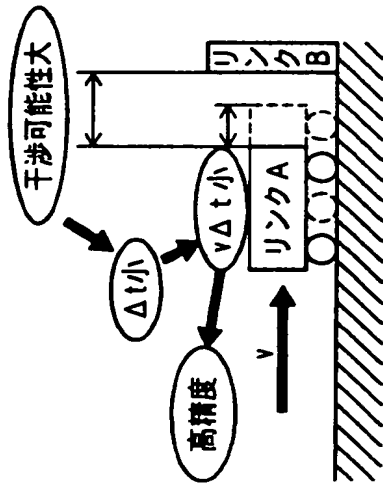
【図 33】



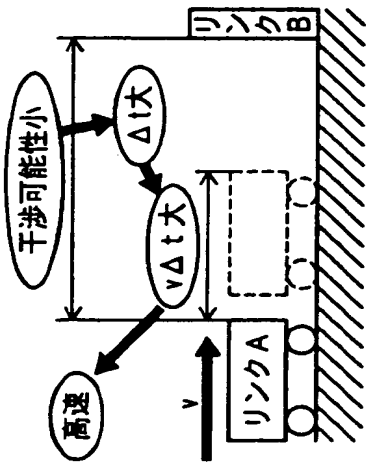
【図34】



【図 3 5】

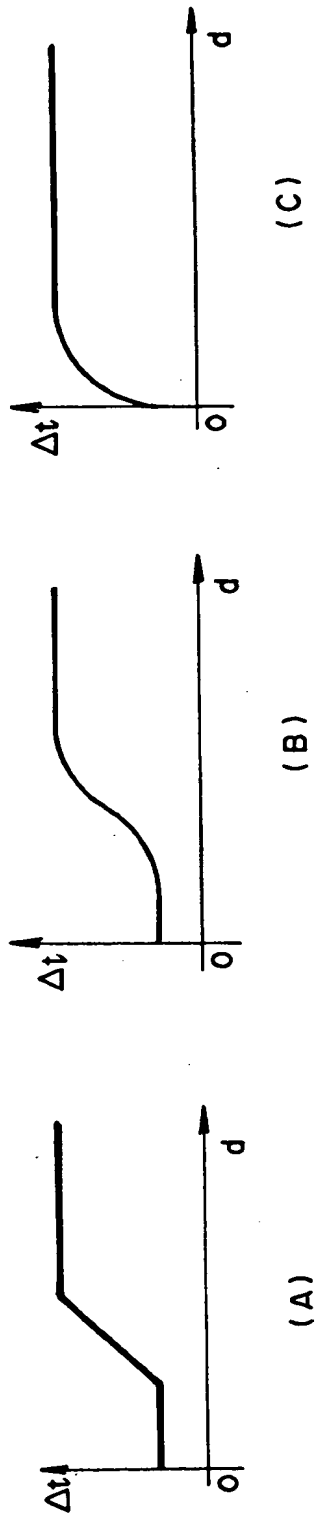


(B)

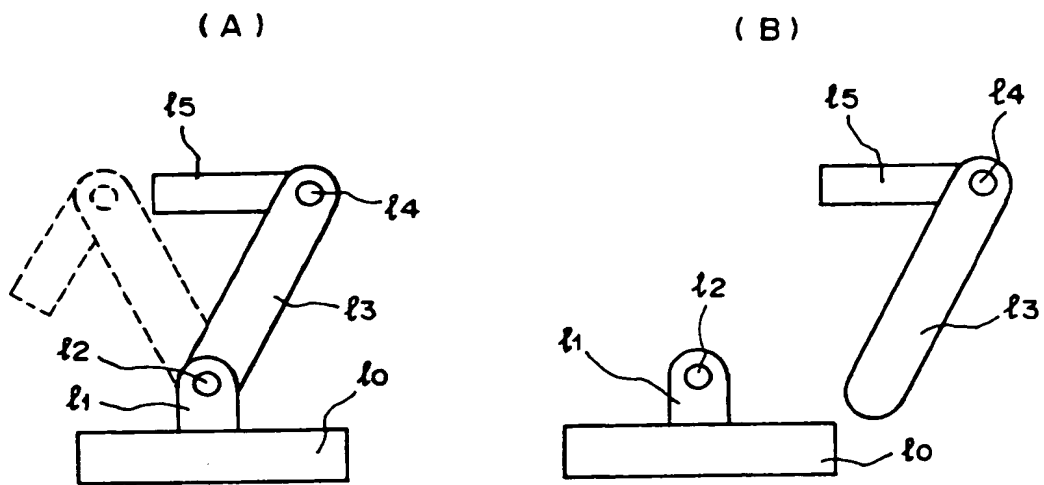


(A)

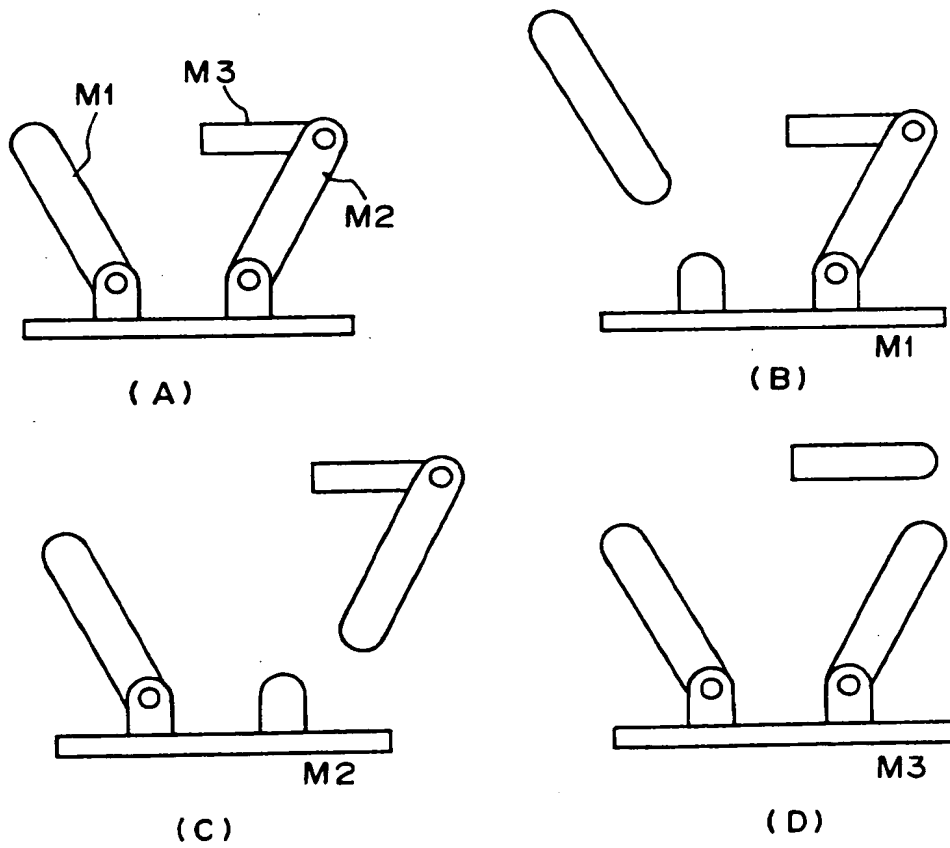
【図 3 6】



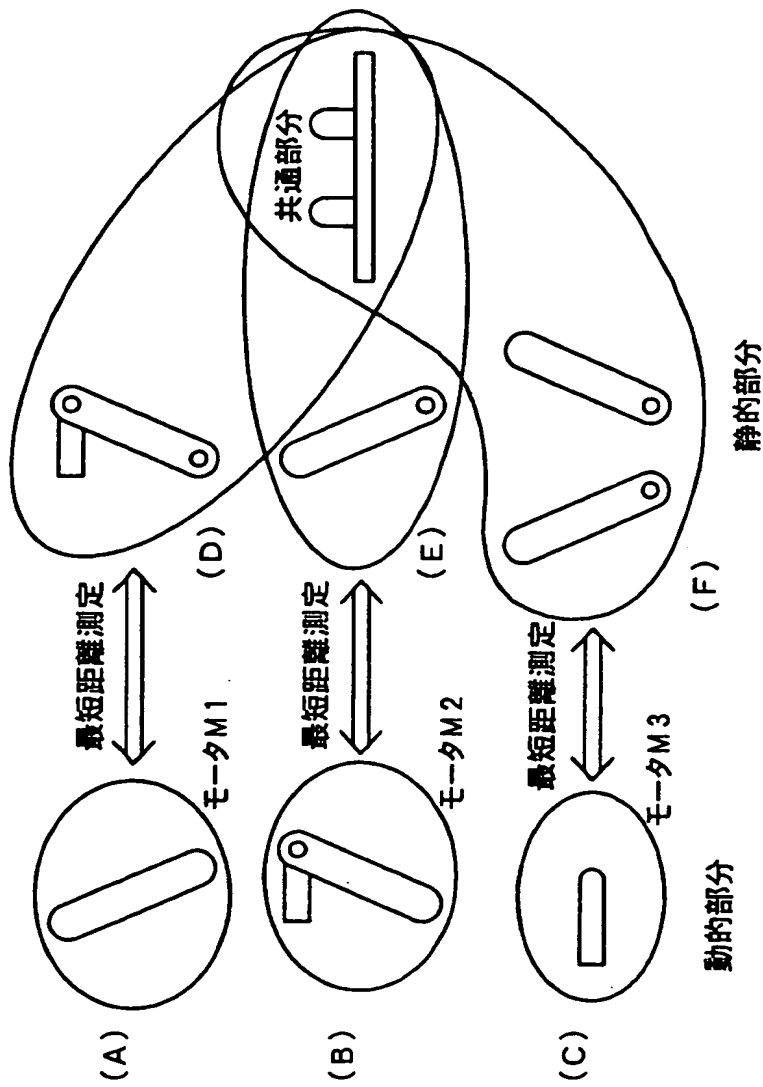
【図 37】



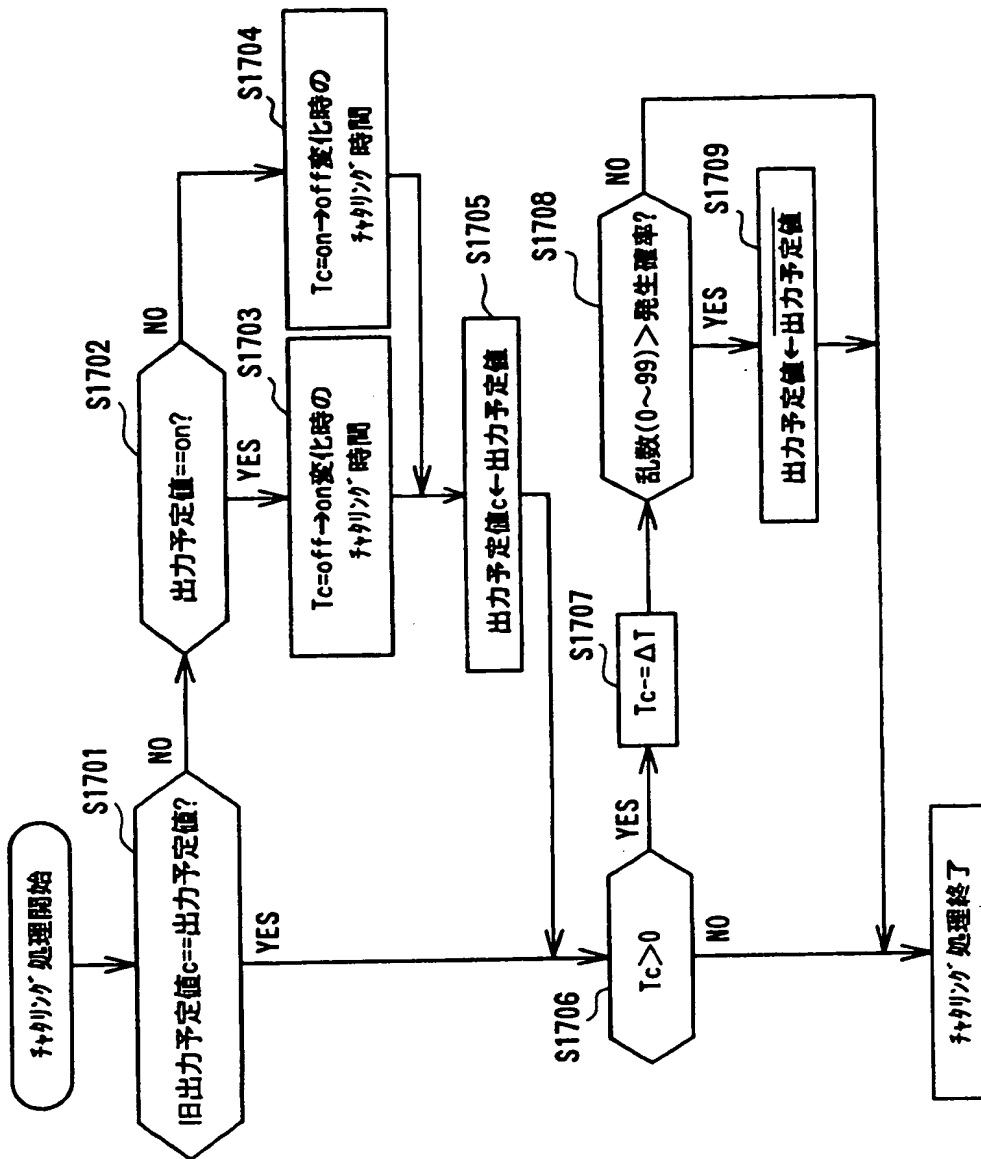
【図 38】



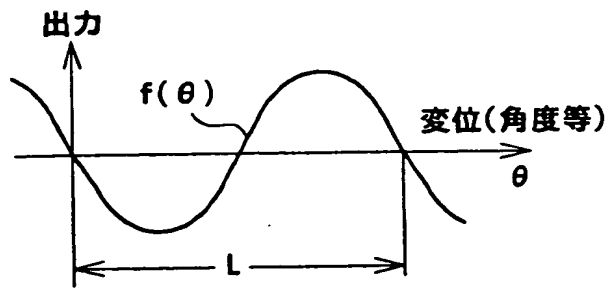
【図 3 9】



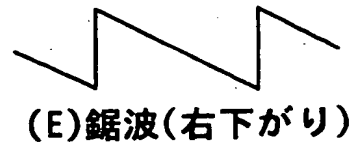
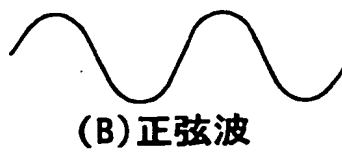
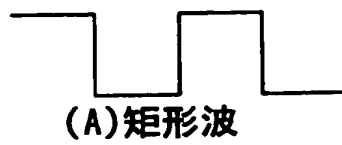
【図 5 2】



【図 5 3】

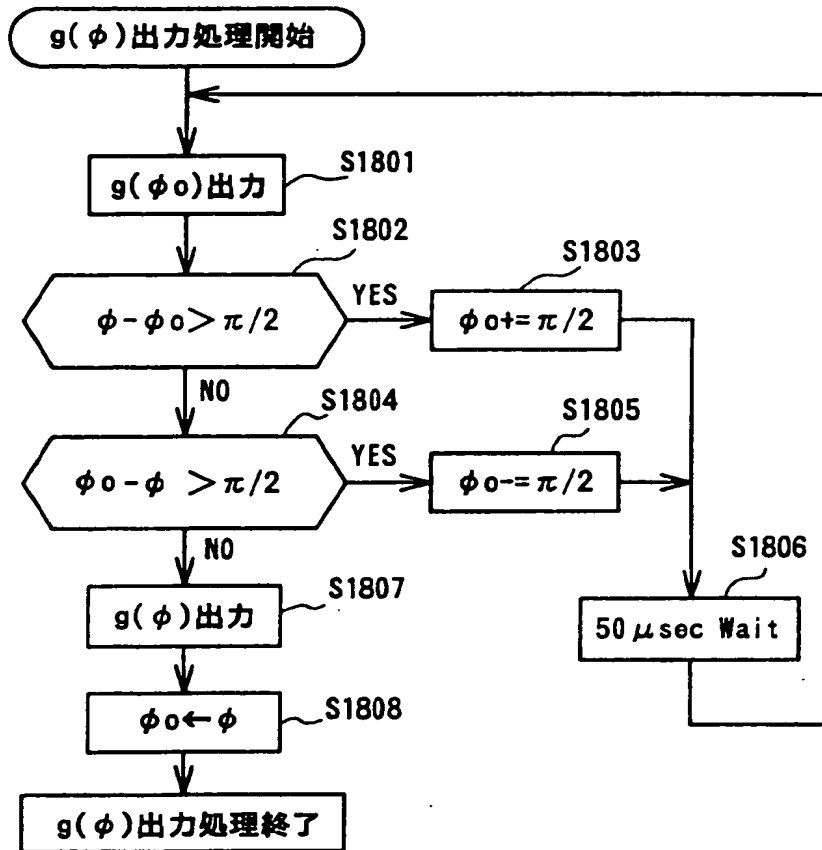


【図 5 4】

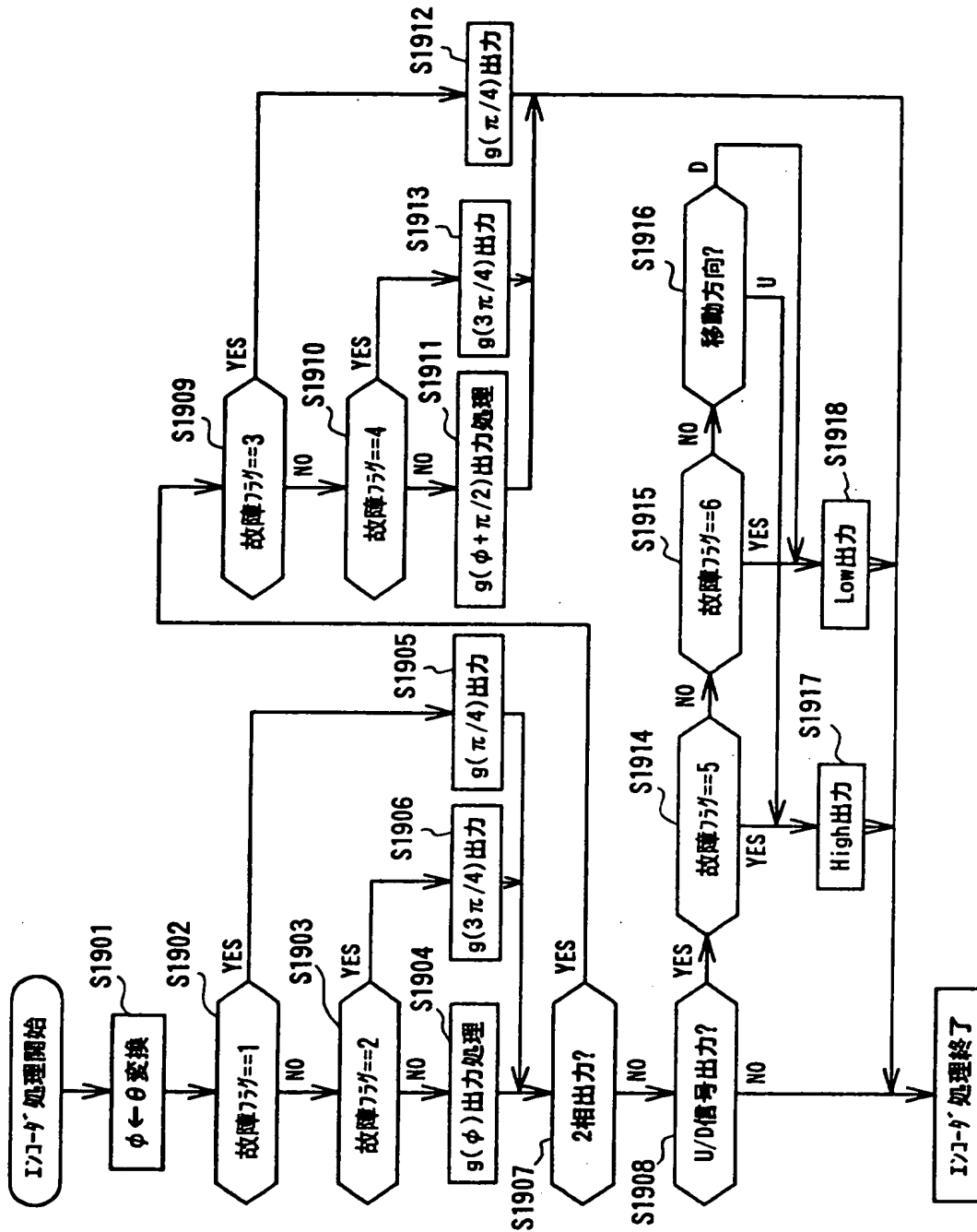




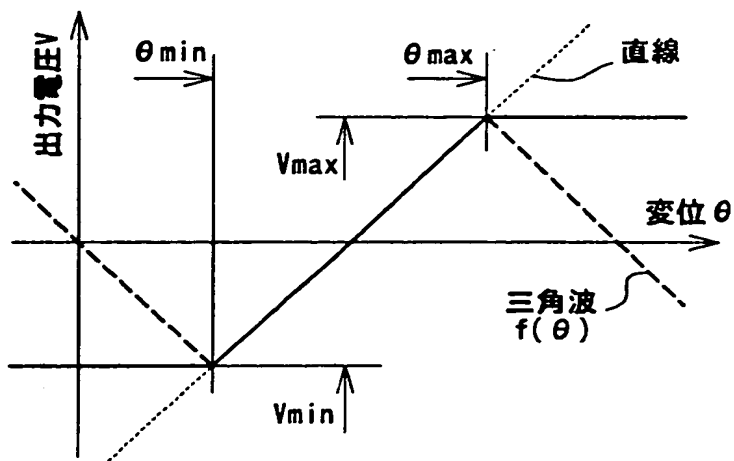
【図 5 5】



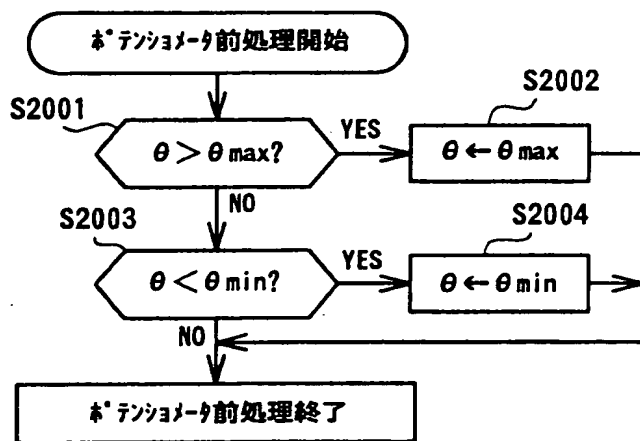
【図 56】



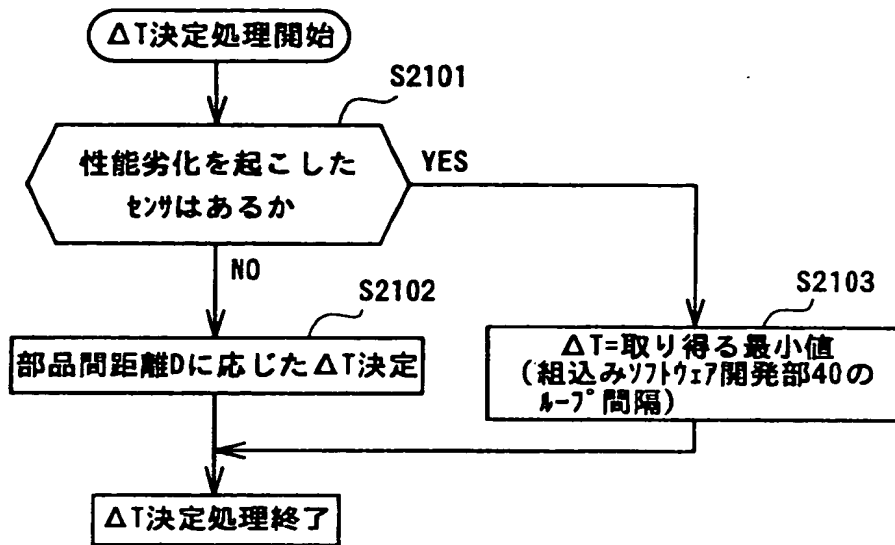
【図 57】



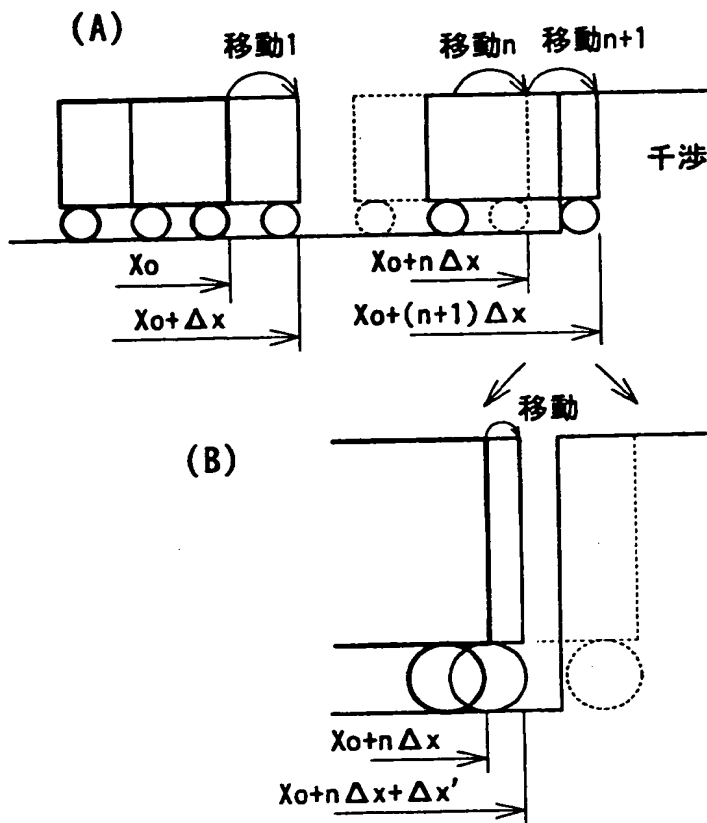
【図 58】



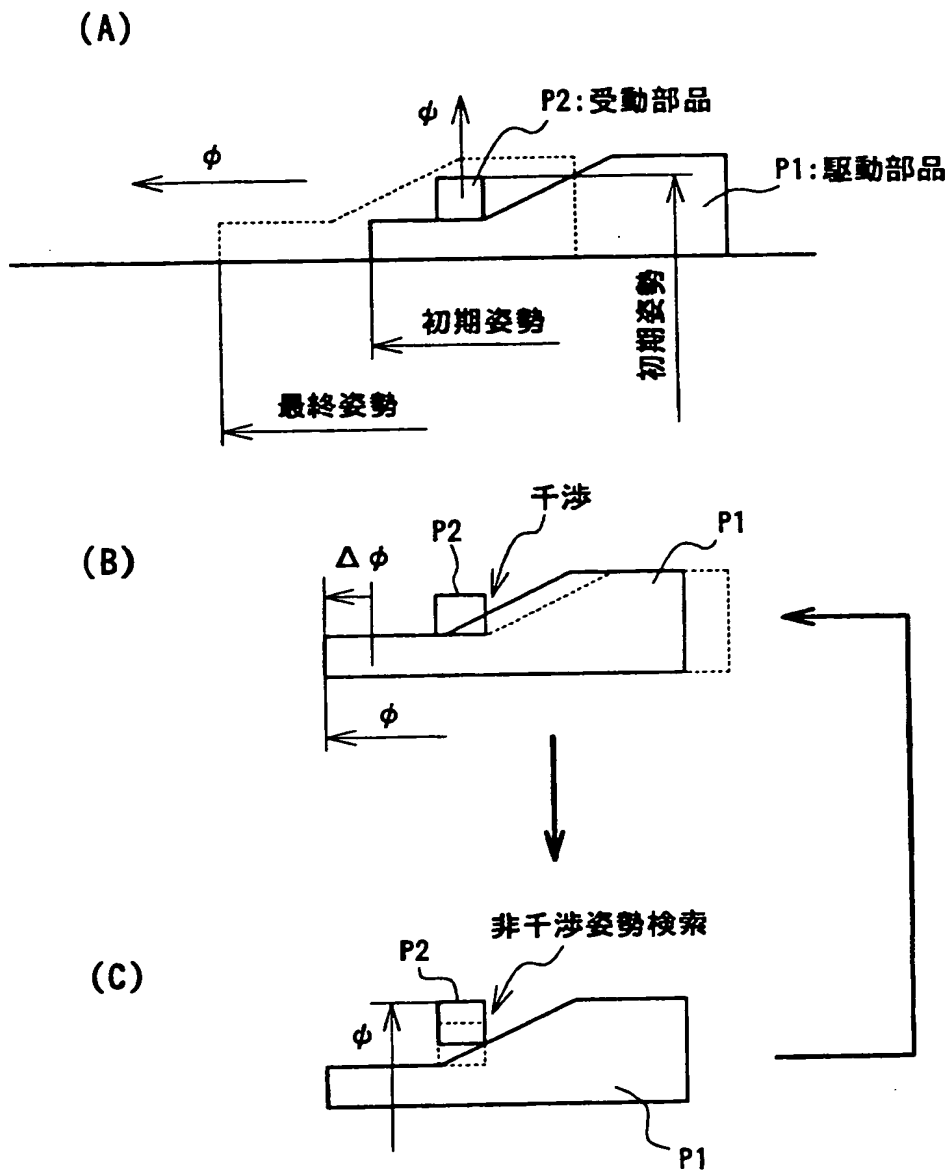
【図 5 9】



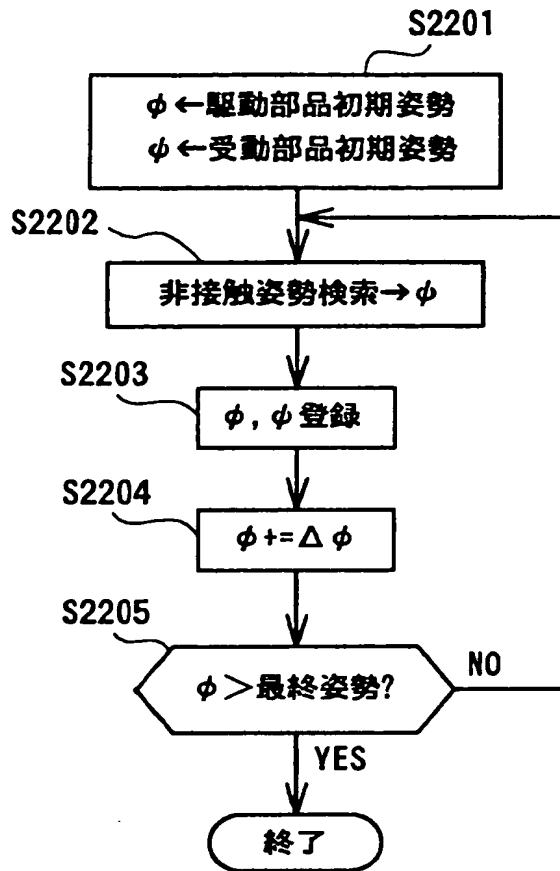
【図 6 0】



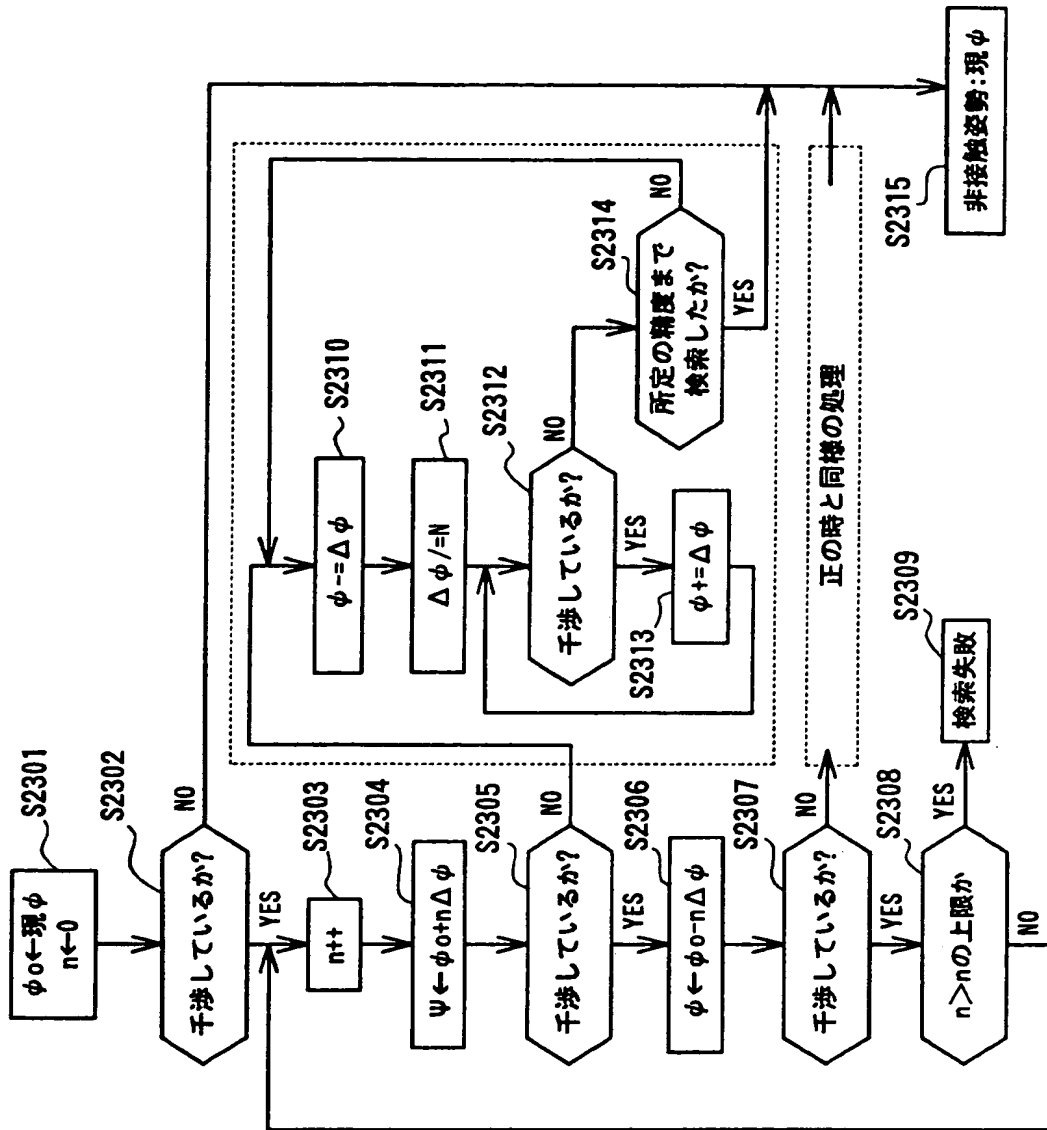
【図 6 1】



【図 6 2】

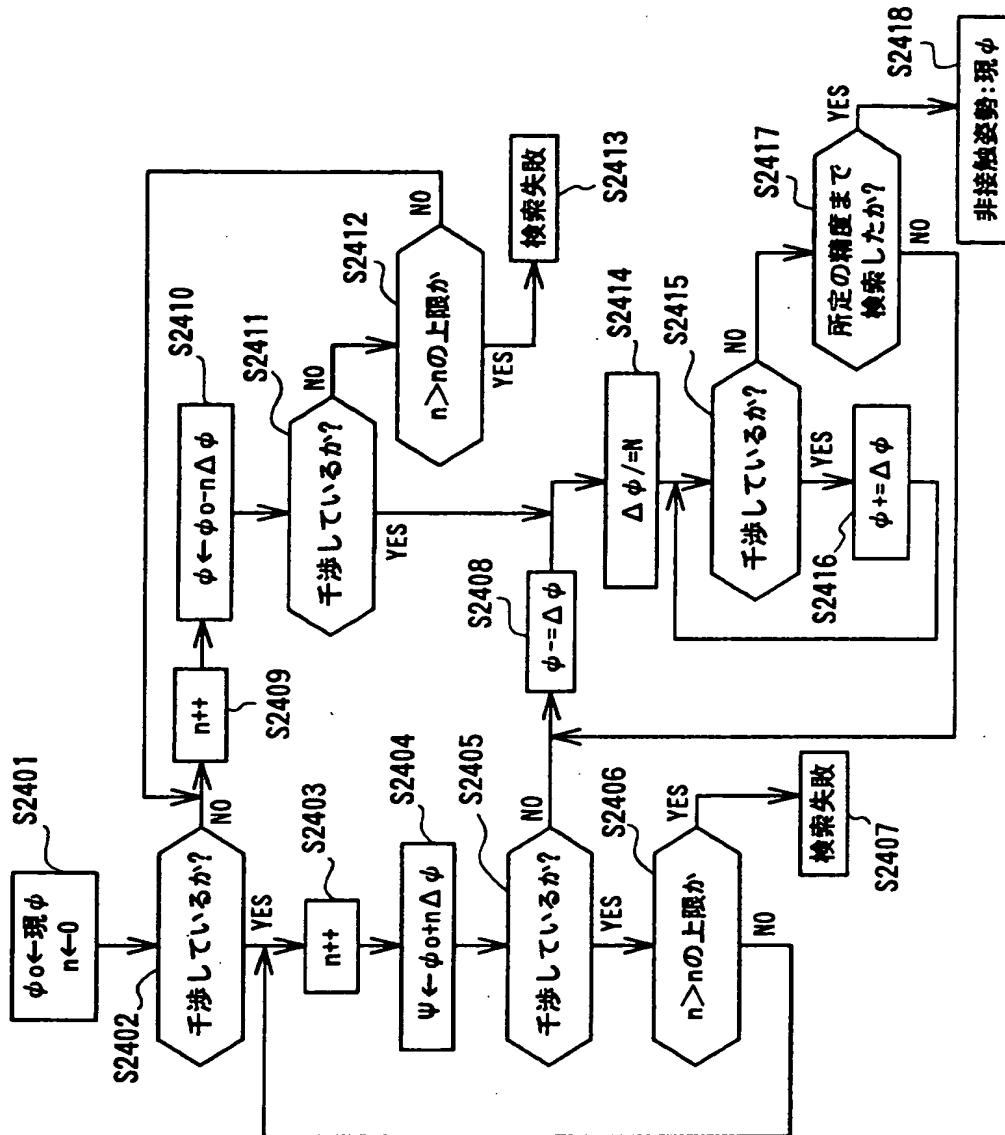


【図 63】

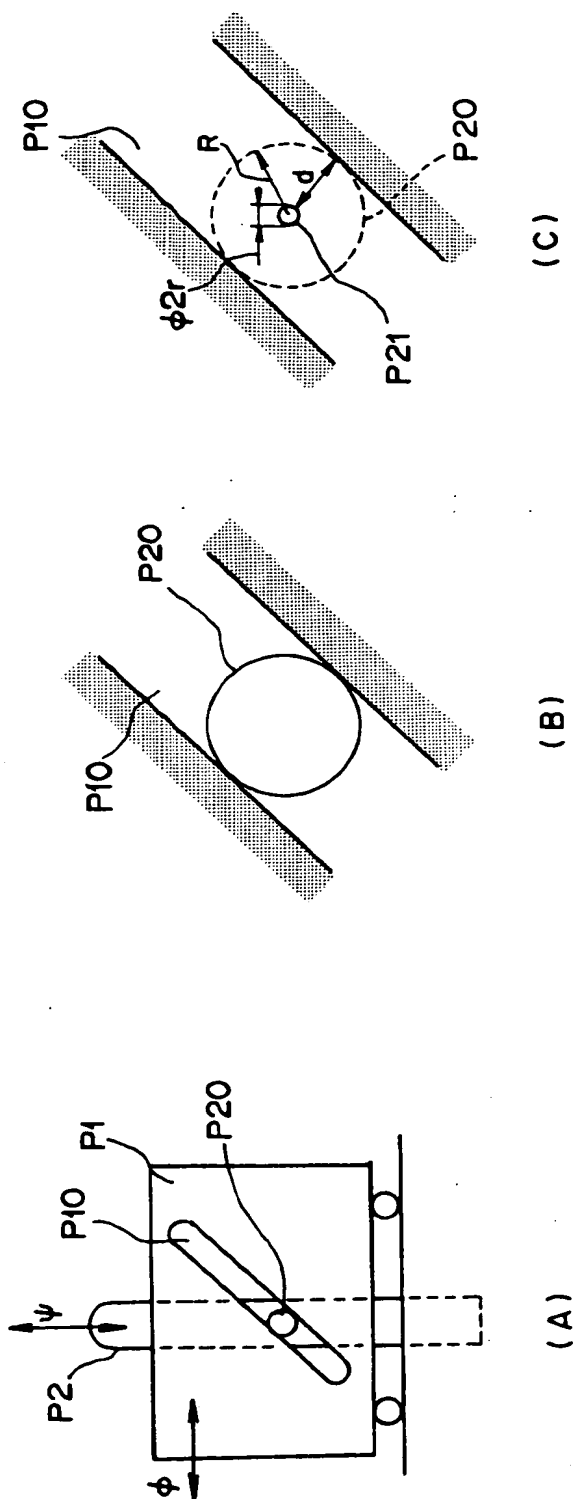




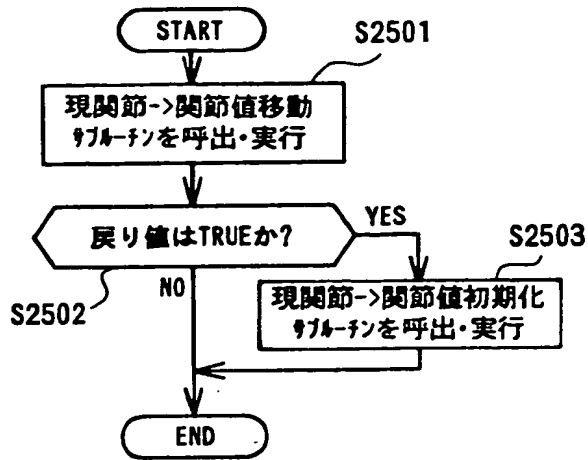
【図 64】



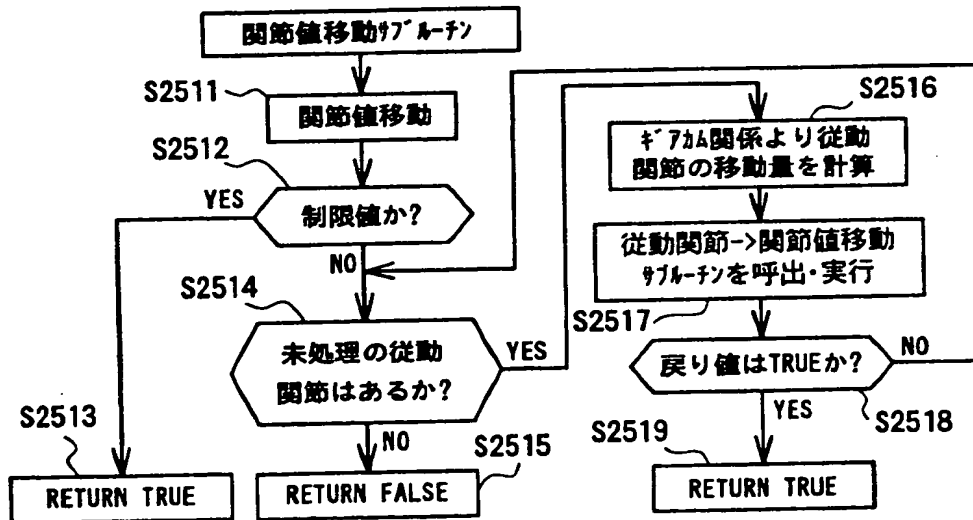
【圖 6 5】



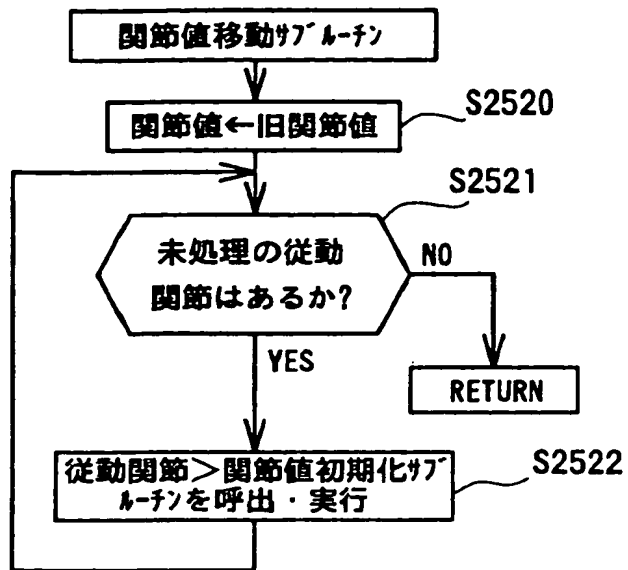
【図 6 6】



【図 6 7】

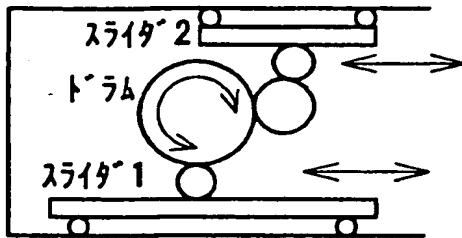


【図 6 8】

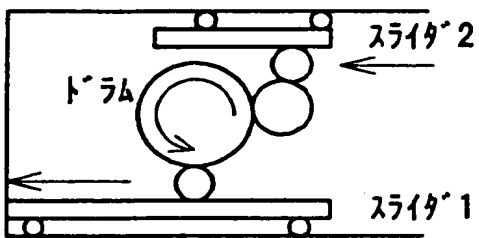


【図 69】

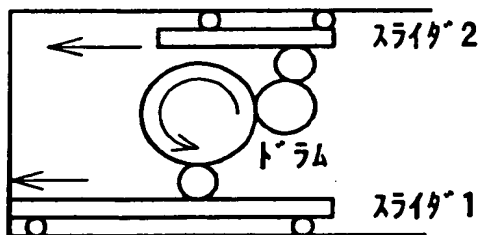
(A)



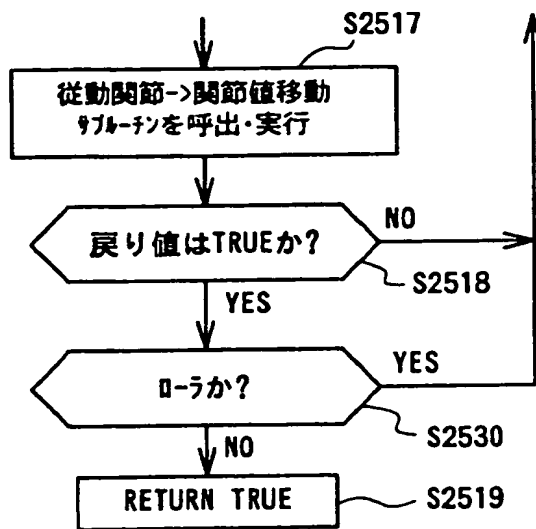
(B)



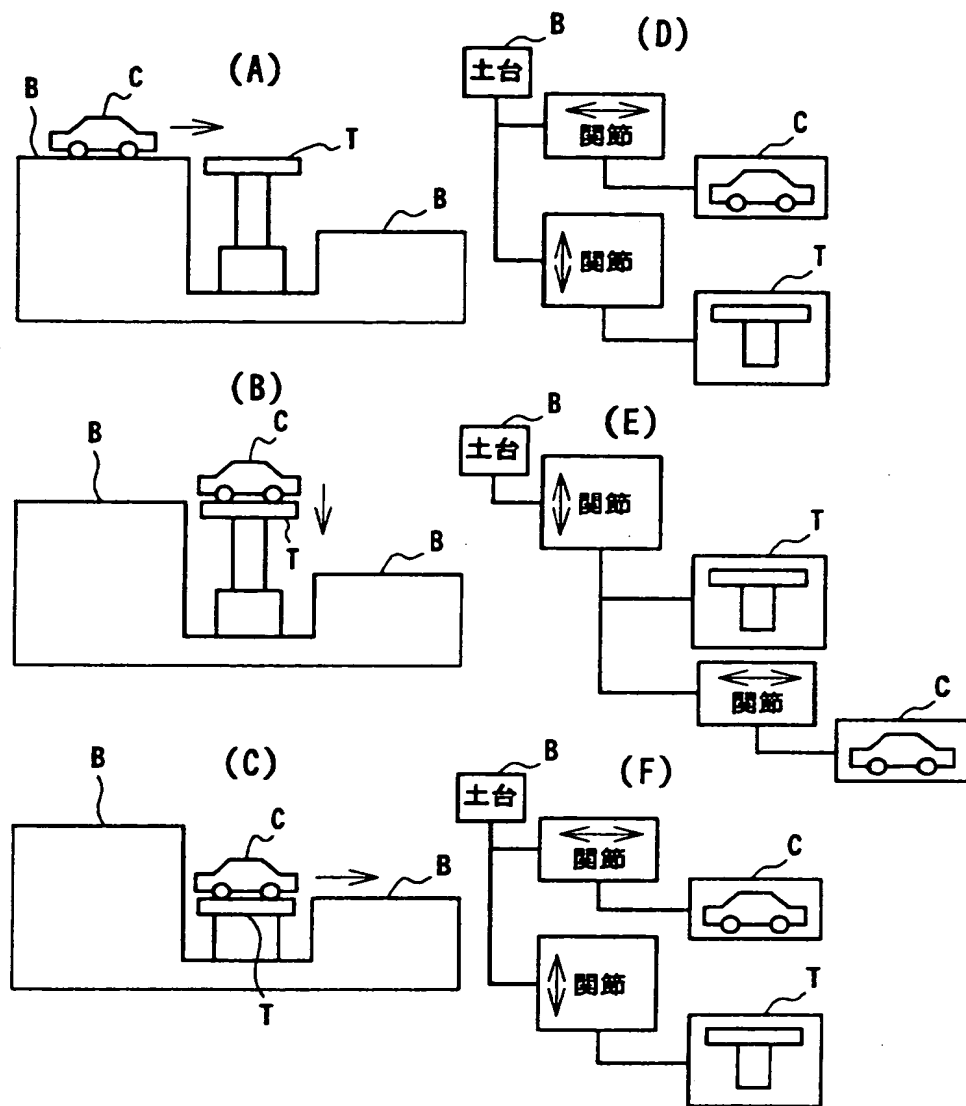
(C)



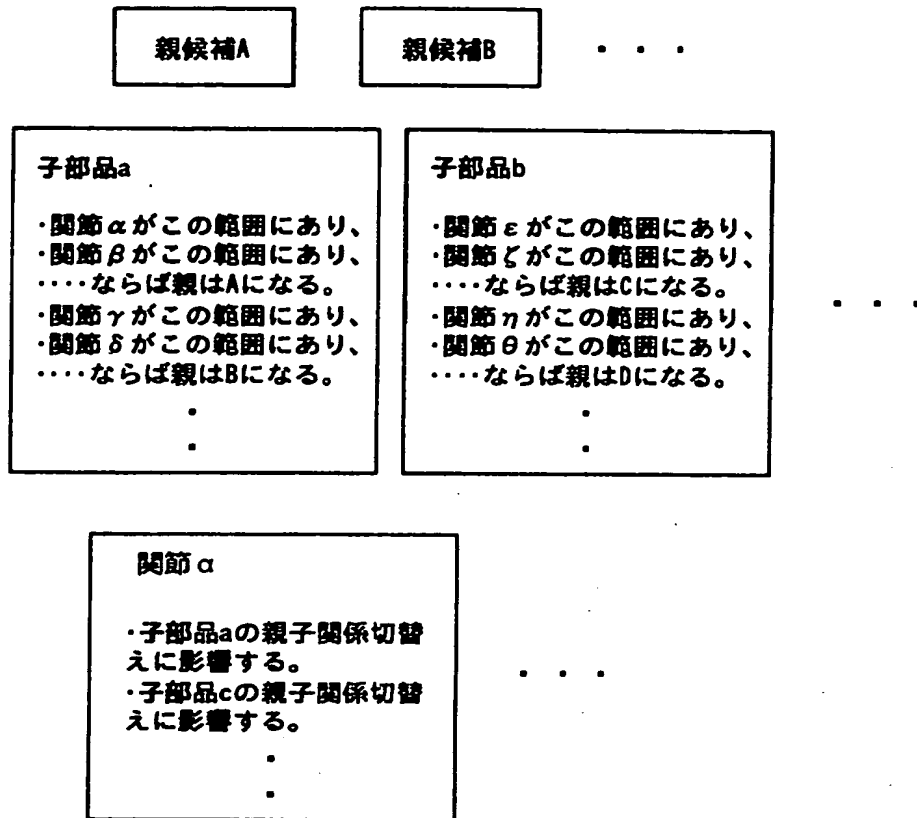
【図 7 0】



【図 71】

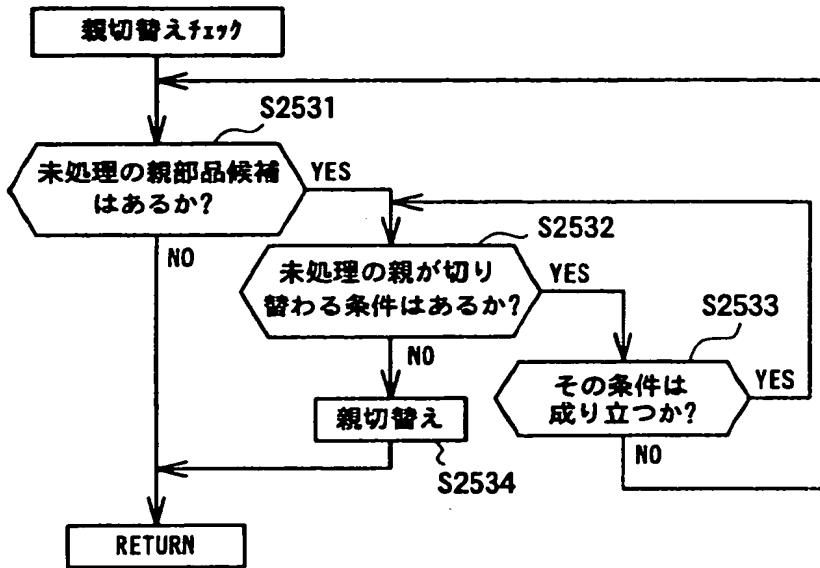


【図 72】

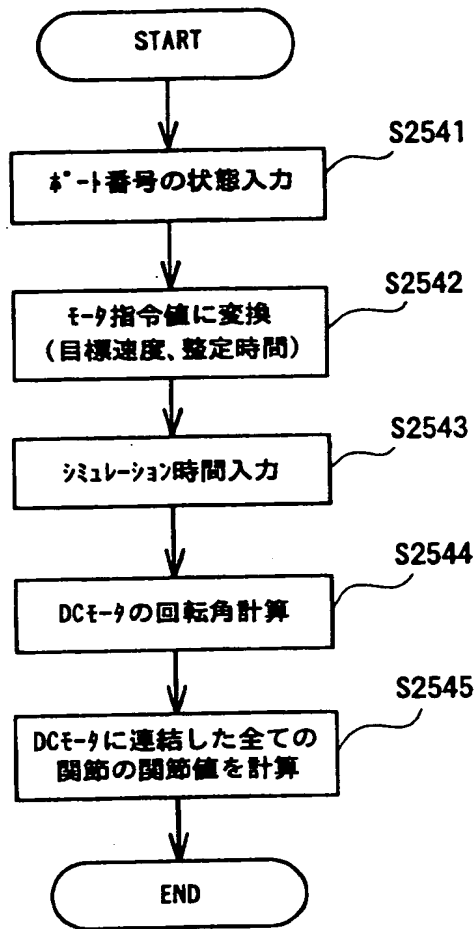




【図 7 3】



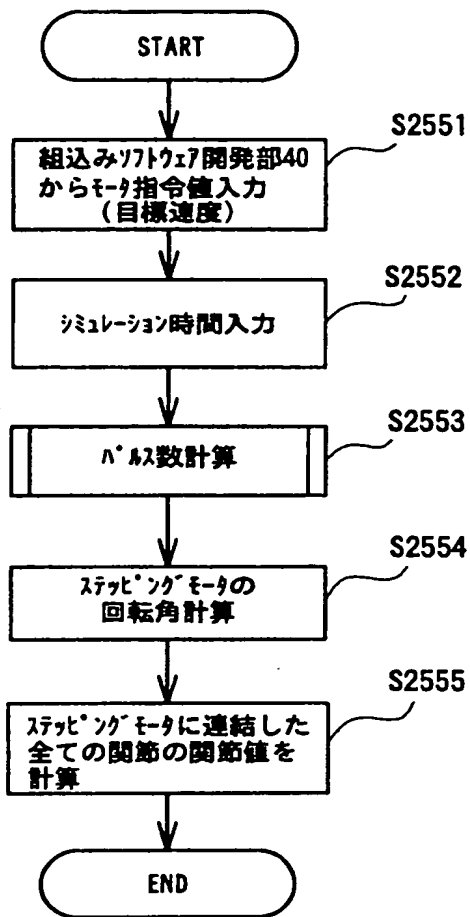
【図 7 4】



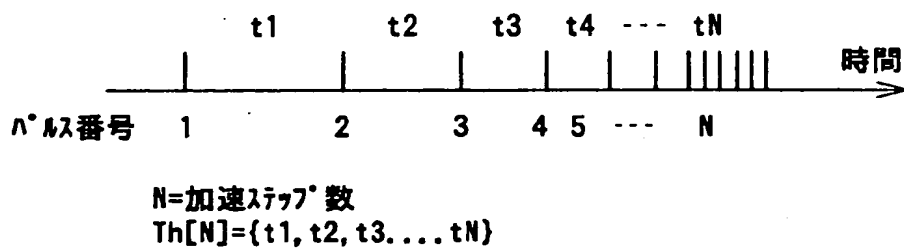
【図 7 5】

	ビット番号		目標速度 [deg / s]	整定時間 [ms]
	11	12		
ビット列に 対応する モータ指令	0	0	0	0
	0	1	+100	500
	1	0	-100	500
	1	1	0	0

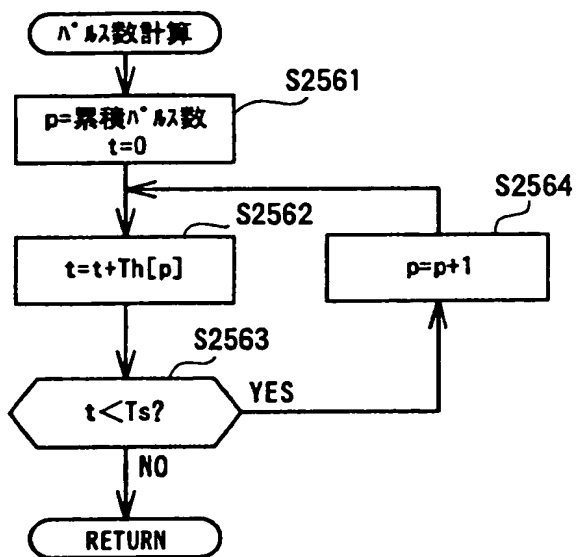
【図 7 6】



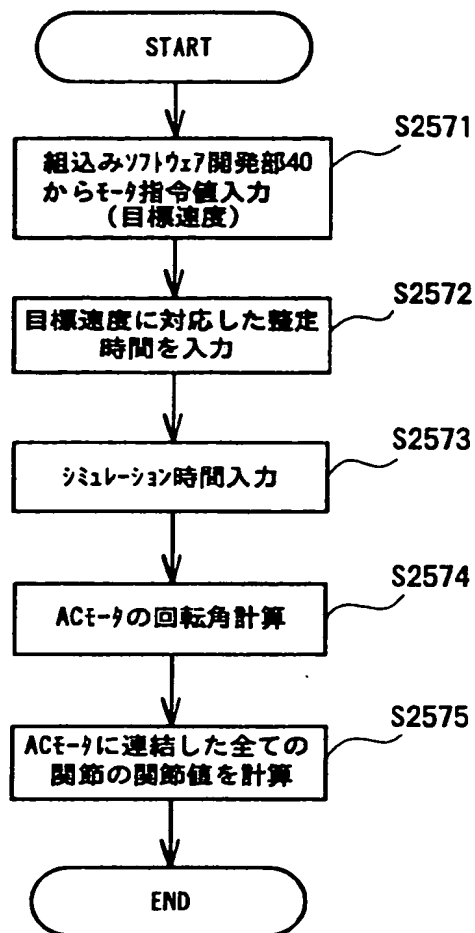
【図 7 7】



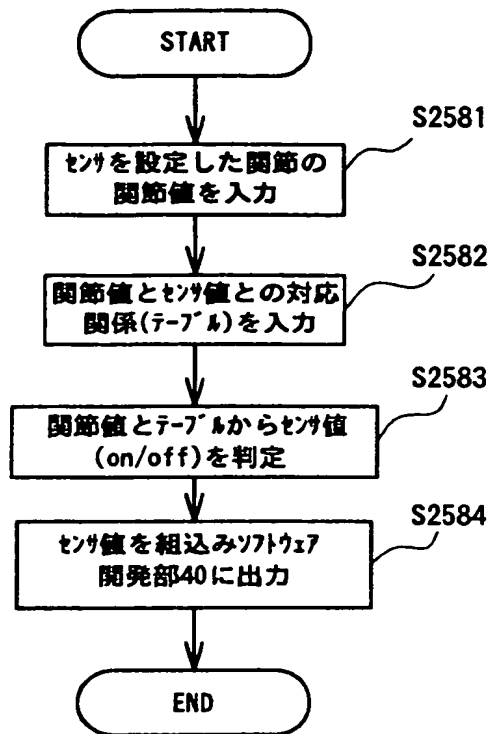
【図 7 8】



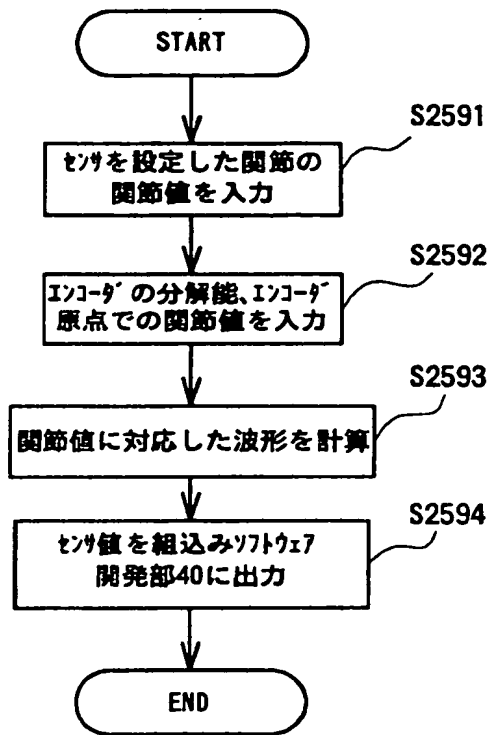
【図 79】



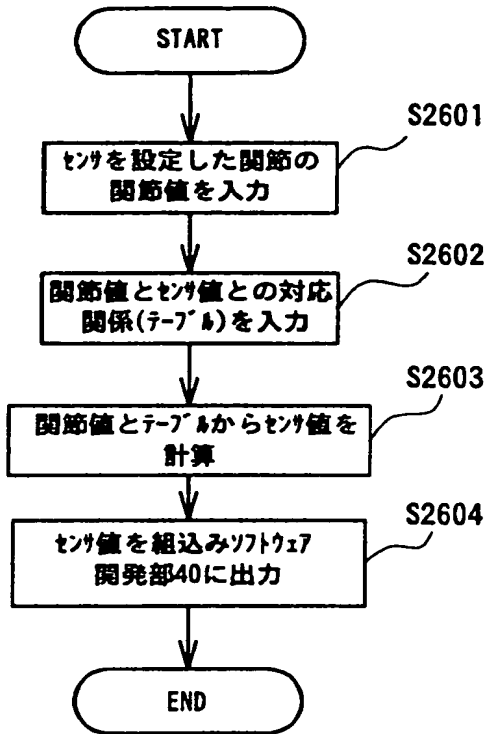
【図 8 0】



【図 81】



【図 8 2】





【書類名】 要約書

【要約】

【課題】機構の設計とその機構を制御するための組込みソフトウェアの開発とをコンカレントに遂行できるようにして、その組込みソフトウェアの開発の効率化をはかり、開発期間の短縮，工数の削減を実現する。

【解決手段】機構を3次元的に設計する機構設計部20と、その機構を3次元機構モデルとして内部に構築され機構の動作をシミュレートする3次元機構モデルシミュレーション部30と、機構の設計と並行して機構の動作を制御するための制御プログラムを開発する組込みソフトウェア開発部40と、設計データを3次元機構モデルに動的に反映させるべく機構設計部20から3次元機構モデルシミュレーション部30へ入力する第1インタフェース部50と、3次元機構モデルシミュレーション部30と組込みソフトウェア開発部40との同期をとりながらデータ送受を行なう第2インタフェース部51，52とがそなわれている。

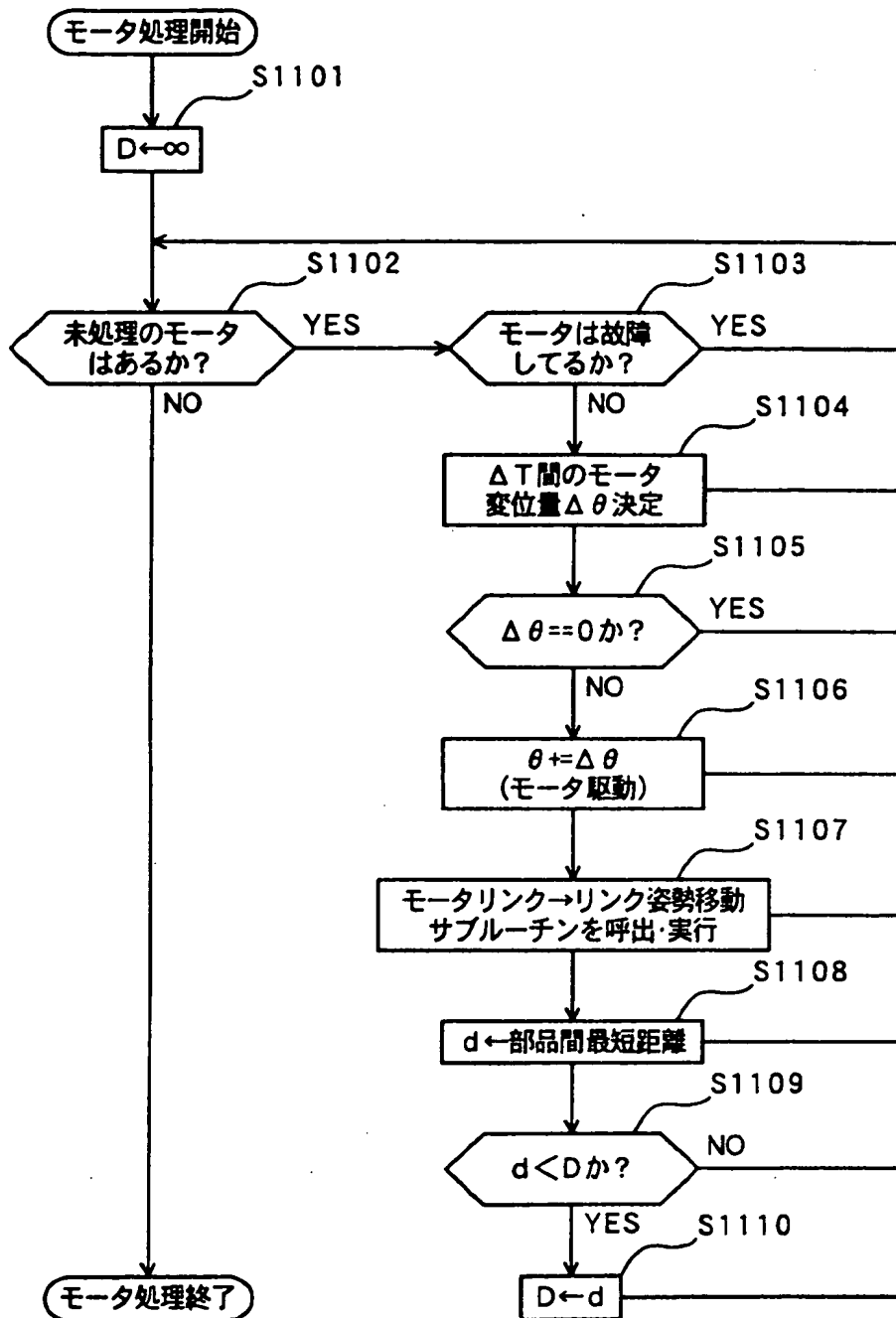
【選択図】 図1

出 願 人 履 歴 情 報

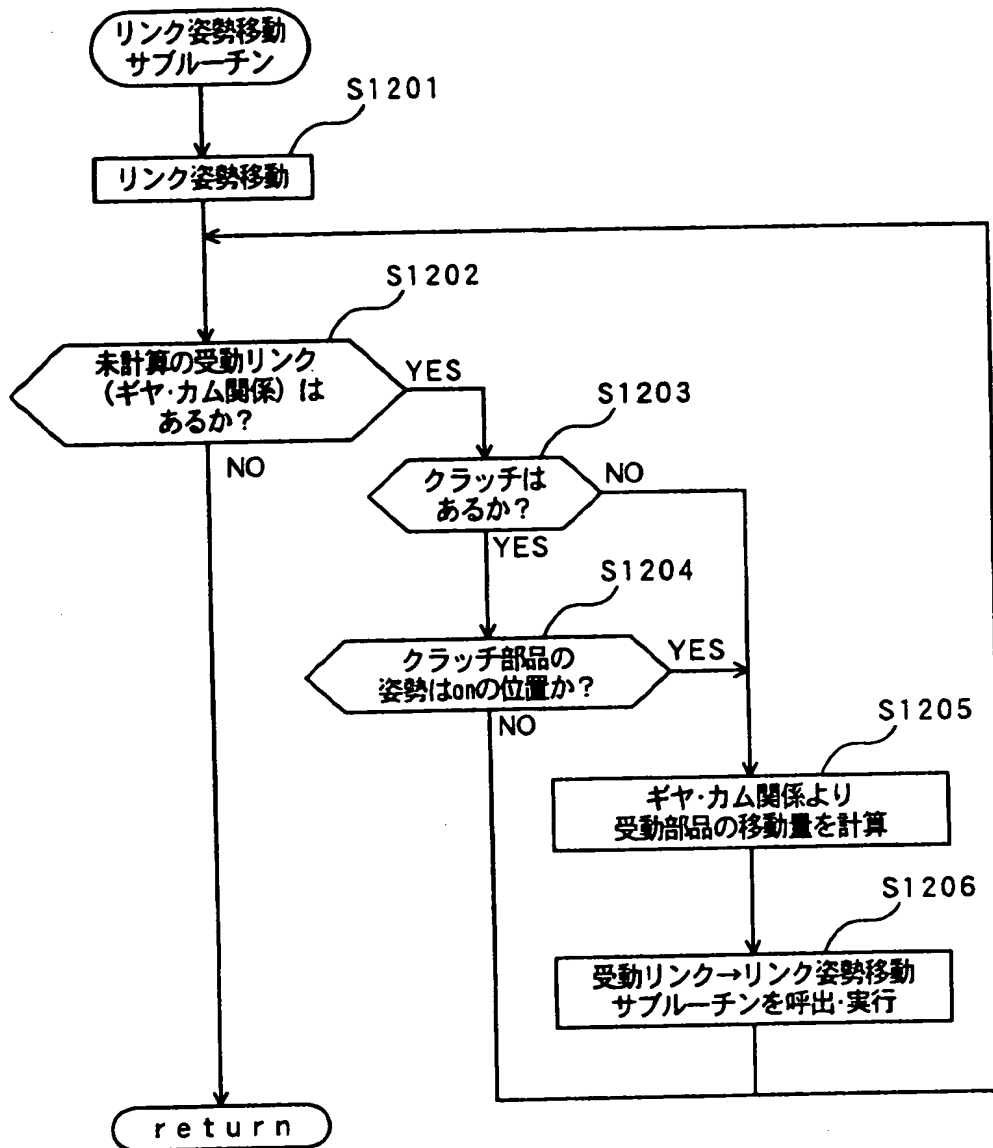
識別番号 [000005223]

1. 変更年月日 1996年 3月26日  
[変更理由] 住所変更  
住 所 神奈川県川崎市中原区上小田中4丁目1番1号  
氏 名 富士通株式会社

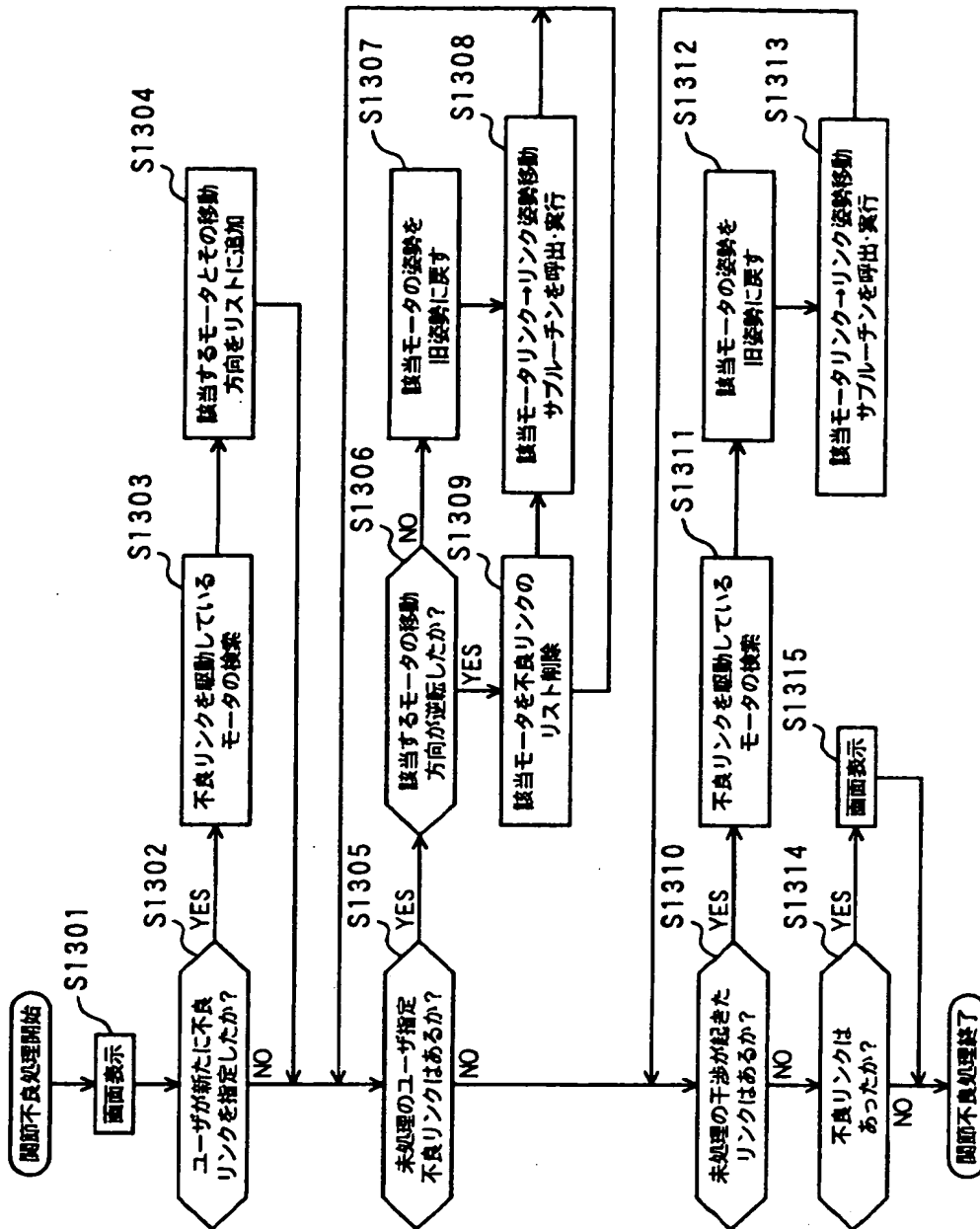
【図 46】



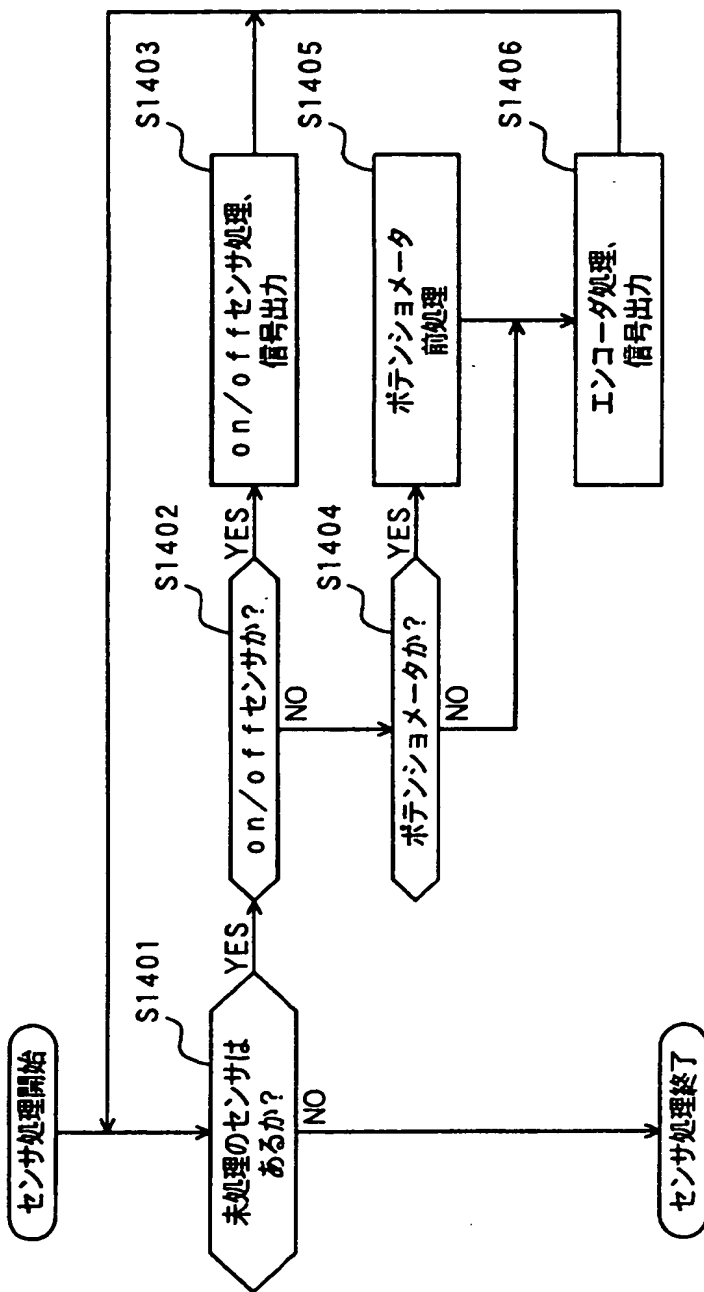
【図 47】



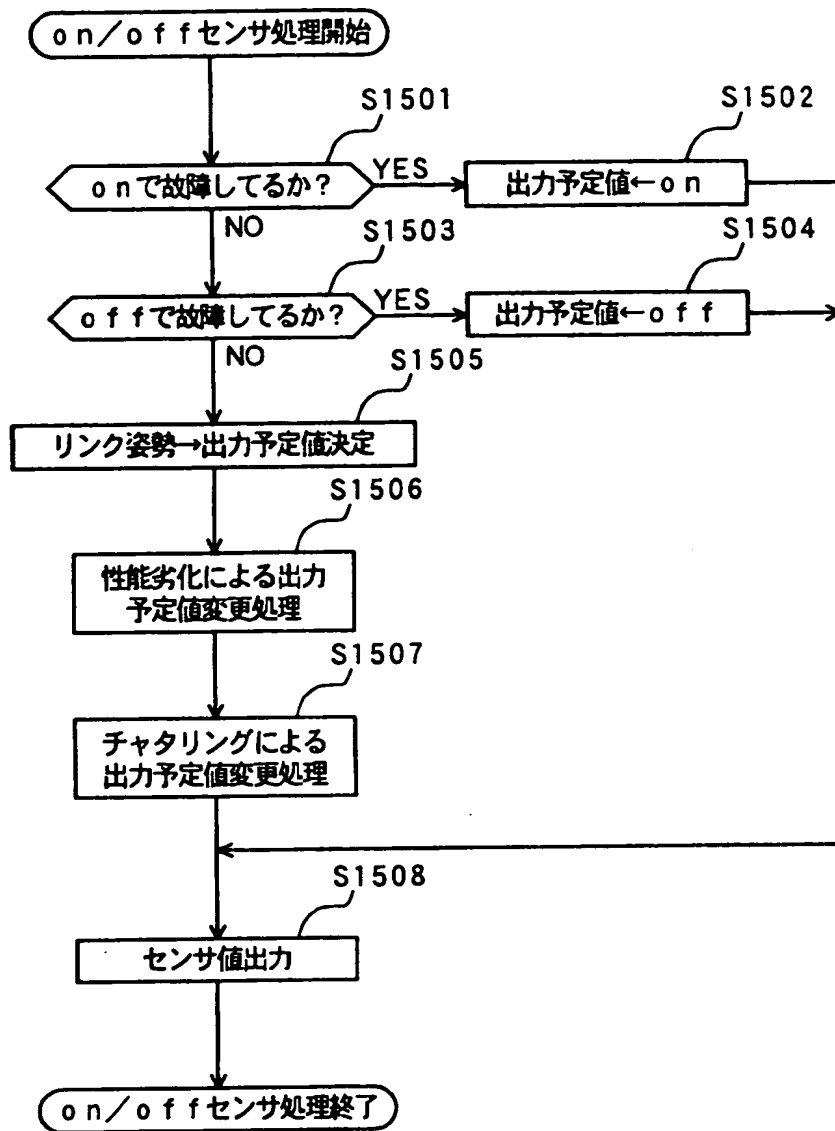
【図 48】



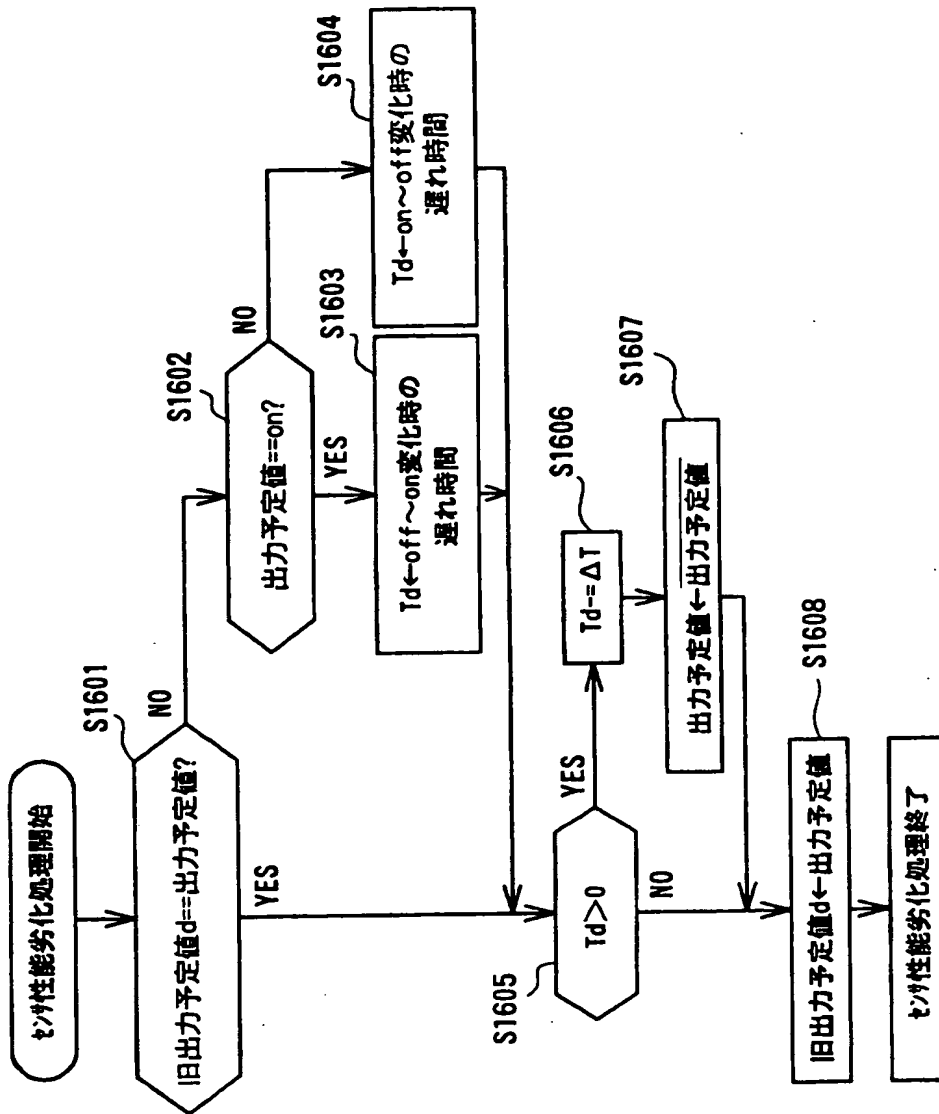
【図 49】



【図 5 0】

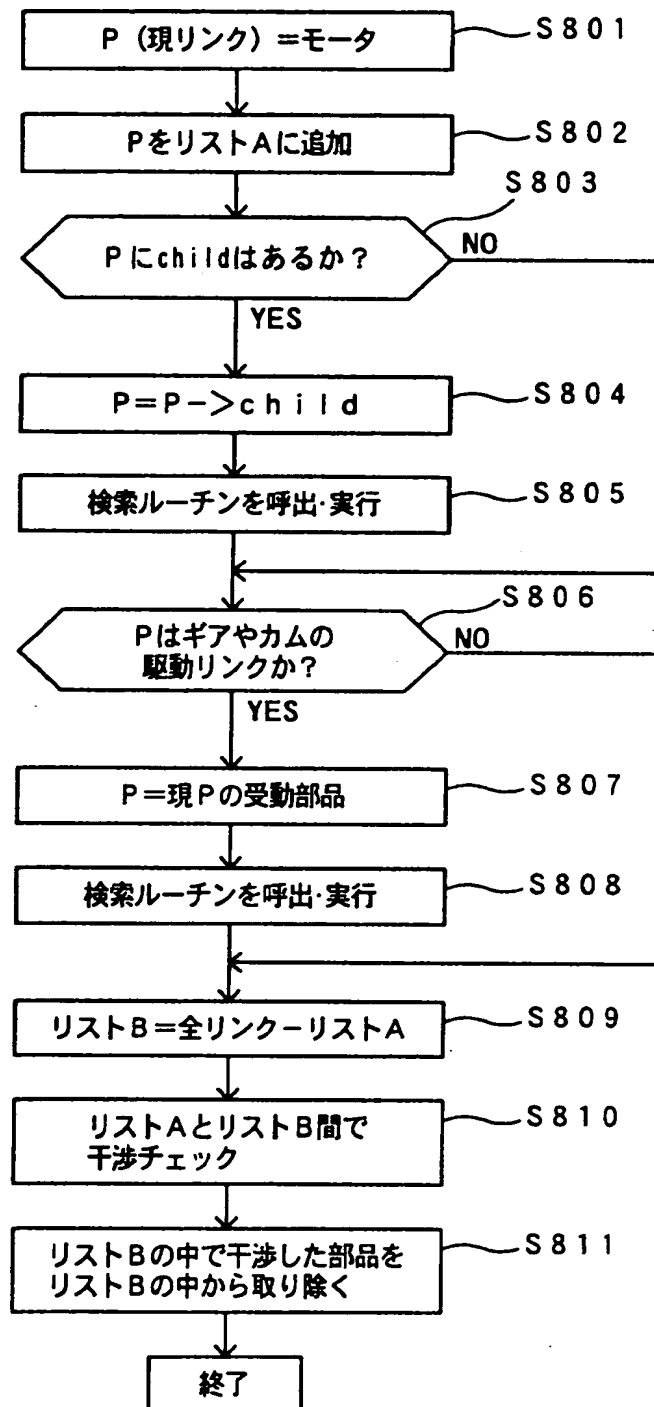


【図 5 1】

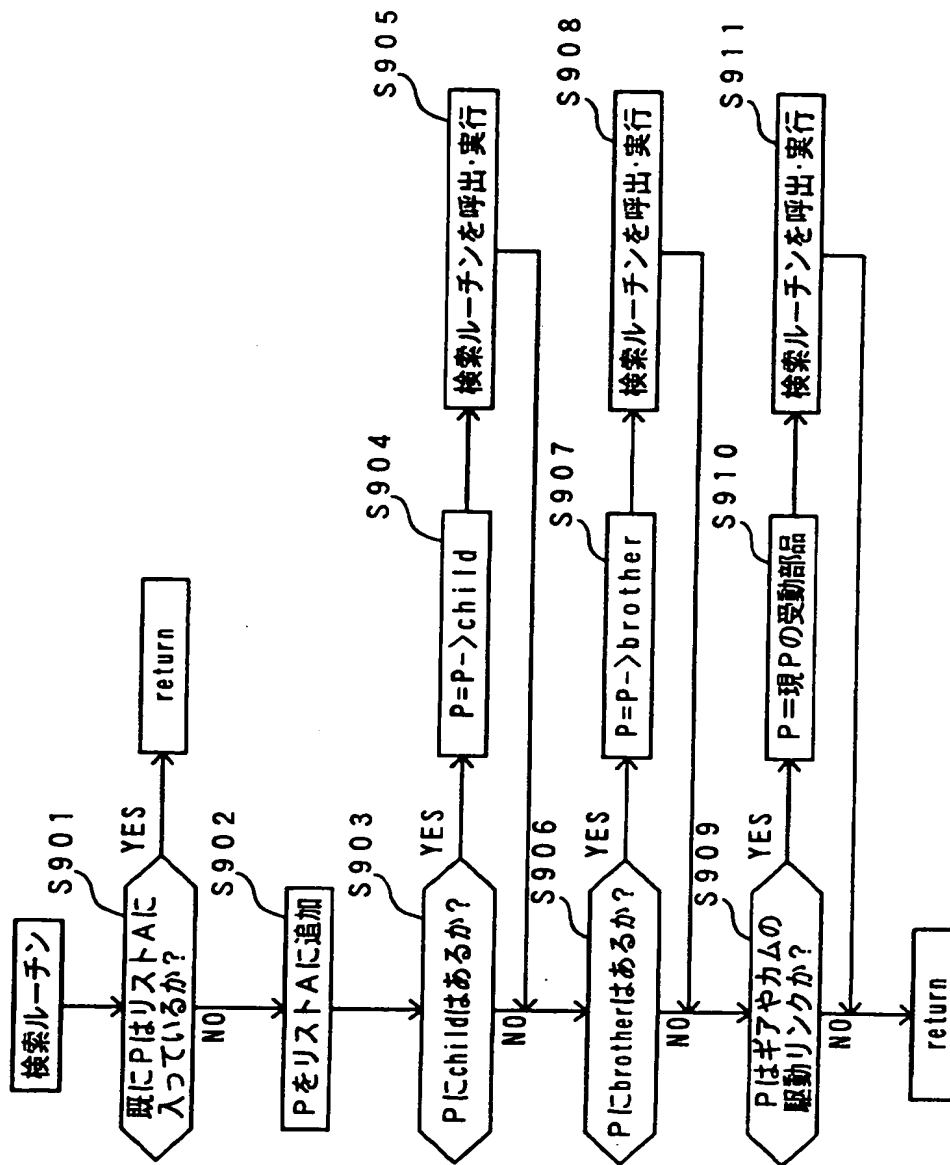




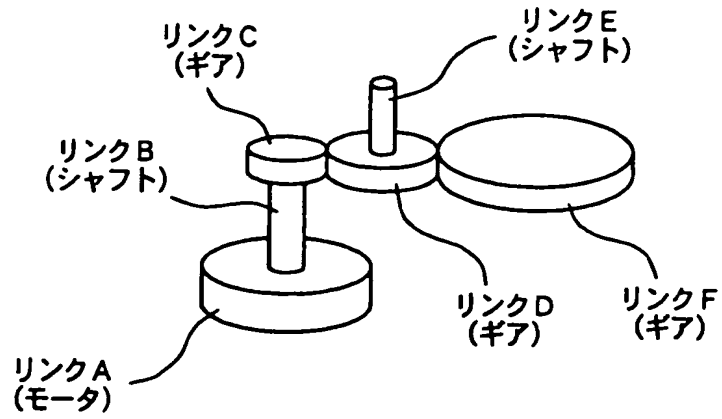
【図 4 0】



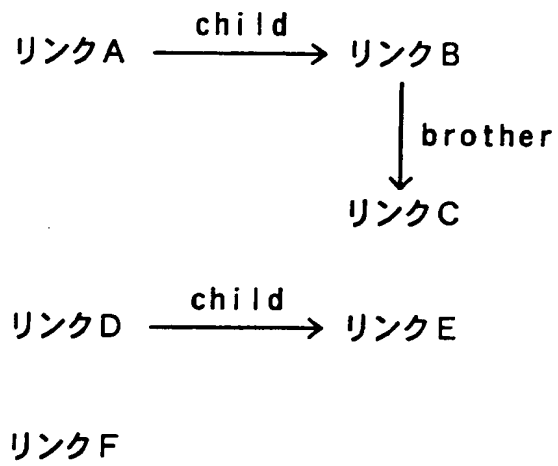
【図 4 1】



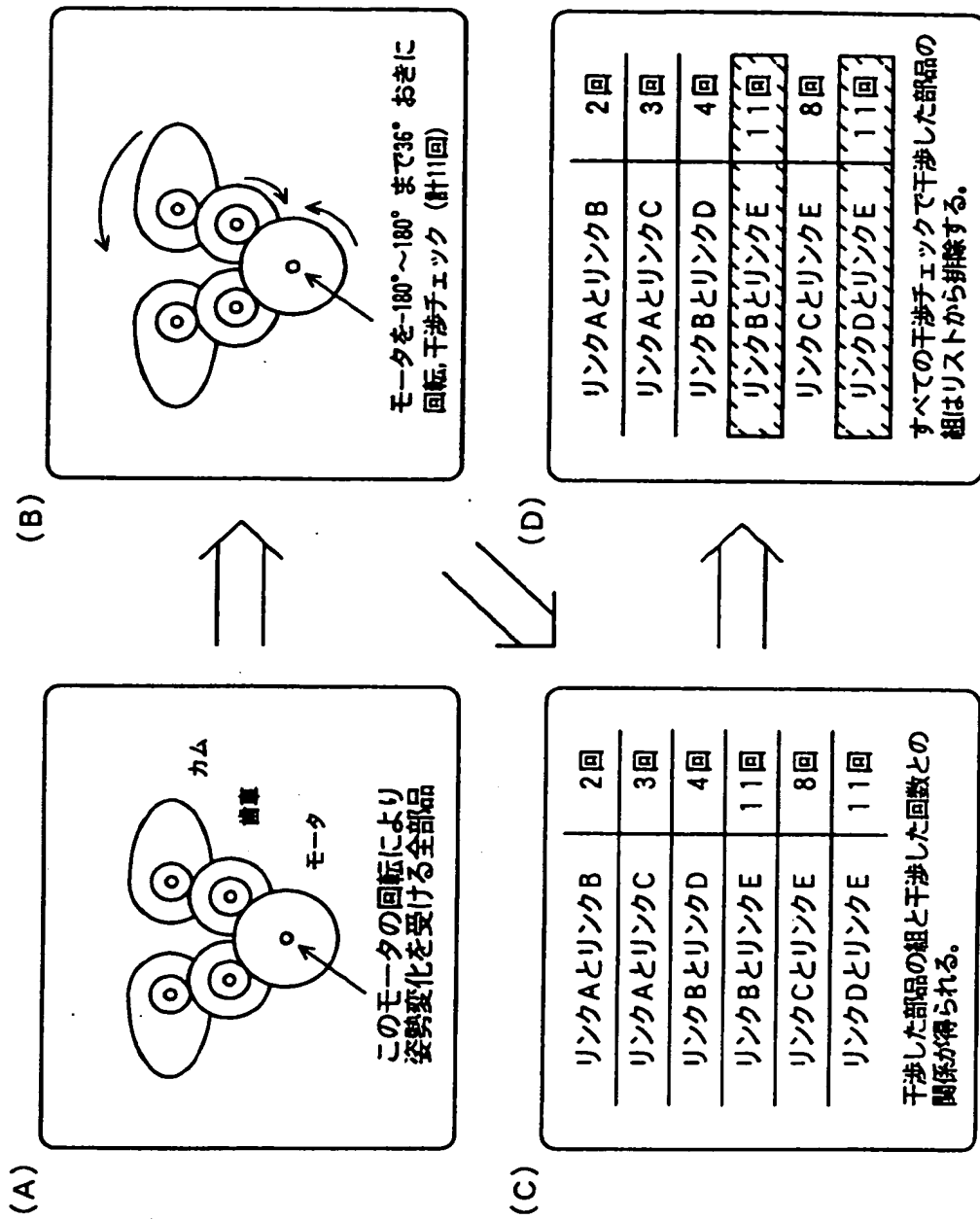
【図 4 2】



【図 4 3】



【図 4 4】



【図 4 5】

